VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza a Syntéza textur

Texture Analysis and Synthesis

2017 Bc. Ondřej Lazarczyk

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Ondřej Lazarczyk**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Analýza a Syntéza textur
Texture Analysis and Synthesis

Jazyk vypracování: čeština

Zásady pro vypracování:

Jedním z problémů při vytváření modelů a následném texturování je obvykle nedostatek vhodných textur s dostatečně velkým rozlišením. Cílem této práce je navrhnout a vytvořit nástroj pro generování textur, který bude využívat vstupní textury menšího rozlišení, a ve výsledku pomocí jejich kombinací vytvářet textury nové s větším rozlišením.

1. Seznamte se s metodami analýzy a generování rozměrných textur z reálných snímků.
2. Aktuálně používané metody pro syntézu textur teoreticky popište, vybrané metody naimplementujte a prakticky srovnejte (např. metody image quilting, parallel controllable texture synthesis apod.). Zaměřte se také na možnost mapování těchto textur na 3D modely.
3. Vytvořte aplikaci, která bude umožňovat pomocí různých metod generovat výsledné textury s větším rozlišením.
4. Výsledek otestujte na sadě snímků nanesením výsledné textury na vhodný model.

Seznam doporučené odborné literatury:

[1] EFROS, Alexei A. a William T. FREEMAN. Image quilting for texture synthesis and transfer. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01 [online]. New York, New York, USA: ACM Press, 2001, s. 341-346 [cit. 2015-09-05]. ISBN 158113374x. Dostupné z: http://people.csail.mit.edu/billf/publications/Image_Quilting.pdf
[2] LEFEBVRE, Sylvain a Hugues HOPPE. Parallel controllable texture synthesis. In: ACM Transactions on Graphics [online]. 2005, 24(3), s. 777-786 [cit. 2015-09-12]. ISSN 07300301. Dostupné z: http://research.microsoft.com/en-us/um/people/hoppe/paratexsyn.pdf

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Martin Němec, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 26. dubna 2017

Ondřej Laraczyk

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Martinu Němcovi, Ph.D za odbornou pomoc a cenné rady při vypracování této diplomové práce.

Abstrakt

Tato práce se zabývá analýzou a syntézou textur ze vzorových obrazů. Syntéza popisuje způsob, kterým je ze snímku nedostačujících rozměrů vytvořena textura požadovaných rozměrů tak, aby výsledek odpovídal definovaným požadavkům. V první části práce jsou popsány textury, jejich dělení používané pro porovnávání vytvořených výsledků a také způsob mapování na povrch objektů. Dále jsou uvedeny a teoreticky popsány algoritmy aktuálně používaných metod pro generování textur. V rámci práce byly naimplementovány a prakticky srovnány vybrané metody z kategorie patch-based. Následující část práce obsahuje popis metod Image Quilting a Graphcuts. Součástí práce je také úprava metody Image Quilting využívající algoritmů pro detekci hran v obraze a úprava metody Graphcuts umožňující její zrychlení. V závěru práce jsou popsány problémy vyskytující se při syntéze textur a navržena jejich možná řešení.

Klíčová slova: počítačová grafika, textura, syntéza textur, patch-based, Image Quilting, Graphcuts, minimální řez, maximální tok, detekce hran

Abstract

This thesis focuses on example-based texture synthesis and analysis. Synthesis describes the way in which an image of required dimensions is created from a texture of insufficient dimensions to correspond to defined requirements. The first part of the thesis describes textures, their categorising used for comparison of generated results and how to map the textures on surfaces of objects. Furthermore, main algorithms of currently used methods for texture synthesis are listed and described. Selected methods from patch-based category were implemented and practically compared within the thesis. The next part of the thesis contains descriptions of Image Quilting and Graphcuts methods. Part of the thesis is a modification of Image Quilting method using algorithms for edge detection in an image and modification of Graphcuts method allowing its acceleration. The conclusion of the thesis deals with difficulties occurring during the texture synthesis and suggestions of their solutions.

Key Words: computer graphics, texture, texture synthesis, patch-based, Image Quilting, Graphcuts, min-cut, max-flow, edge detection

Obsah

Seznam použitých zkratek a symbolů

BSD	–	Berkeley Software Distribution
FFT	–	Fast Fourier Transform
GPL	–	General Public License
MRF	–	Markov Random Field
SSD	–	Sum of Squared Differences

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době umožňuje vývoj grafických karet realističtější zobrazení ve všech odvětvích počítačové grafiky zejména ve filmovém a herním průmyslu. Jednou z mnoha částí, které mají podíl na realističnosti, jsou textury. Větší výkon a pamět grafických karet umožňuje využití textur o větším rozlišení, které obsahují více detailů. Při používání textur se vyskytuje mnoho různých problémů. Jedním z těchto problémů je nedostatečné rozlišení textury pro model, na který je potřeba texturu nanést. V případě vyskytnutí takového problému existuje více možností jak jej vyřešit. Jedním z možných řešení je opakování dané textury. Avšak při tomto řešení je na první dojem znatelné skládání stejné textury vedle sebe a to způsobuje nerealisticky vypadající povrch objektu. Dále je ve většině případů znatelný přechod mezi jednotlivými opakováními textury. Viditelné přechody lze vyřešit vytvořením textury, jejíž okraje na sebe navazují. Toto řešení vyžaduje značnou zručnost, znalost postupů pro její vytvoření a je časově náročné.

Jedním z efektivních řešení zmíněných problémů je syntéza textur využívající textury o nedostatečném rozlišení pro vytvoření textury většího rozlišení. Syntéza textur umožňuje generování textur ve větším rozlišení, které vypadají velice podobně jako vzorová textura o menším rozlišení. Při generování je kladen důraz na vytvoření textury tak, aby nebylo znatelné vytvoření nové textury opakováním původní textury. Další informace o analýze a syntéze textur je možné nalézt v článcích [1] a [2].

V této práci jsou nejdříve popsány textury a jejich dělení. Poté je popsán způsob mapování textur na povrchy objektů, který je následován popisem aktuálně používaných metod pro syntézu textur. Z popsaných metod jsou zvoleny metody, které jsou teoreticky popsány. První vybranou metodou je Image Quilting [3] a druhou je metoda Graphcuts [4]. V případě metody Image Quilting je také popsána a otestována úprava využívající detekci hran v obraze. Dále je otestována úprava pro metodu Graphcuts umožňující její zrychlení. Výsledky, kterých tyto metody dosahují, jsou poté vyhodnoceny a srovnány. Na konci práce jsou popsány úpravy, které by mohly vést k řešení problémů vyskytujících se u naimplementovaných metod.

1 Textury

Obraz je množinou bodů Ω. Souřadnice jednotlivých bodů jsou označovány (x, y). Pokud jsou rozměry obrazu M a N je možné zapsat tuto množinu jako $\Omega = \{(x, y)|x = 0, 1, 2 \ldots, M - 1; y = 0, 1, 2 \ldots, N - 1\}$. Jednotlivé body jsou označovány jako pixely. Pixely obrazu mohou uchovávat různé informace. Například barvu pixelu nebo jinou potřebnou hodnotu. Definice obrazu je založena na definici z textu [5].

Barva pixelů může být určena třísložkovým nebo čtyřsložkovým vektorem. U čtyřsložkového vektoru je k intenzitám barev přidána hodnota průhlednosti, tato poslední složka vektoru je označována zkratkou a.

Dvourozměrná difúzní textura je obraz, jehož pixely obsahují informaci o barvě. Barva je určena vektorem o třech složkách, které udávají intenzitu červené, zelené a modré barvy. V následujícím textu jsou pro barevné složky používány zkratky r, g, b (červená, zelená, modrá). Základním prvkem textur je texel. Texel textury odpovídá pixelu na stejné pozici. Pro zjednodušení je při použití slova textura v textu této práce myšlena dvourozměrná difúzní textura. Tato práce je zaměřena pouze na dvourozměrné difúzní textury.

Textury obsahují různé vlastnosti povrchu. Informace o těchto vlastnostech obsažené v texturách mají zásadní vliv na výsledný vzhled povrchu objektu. Mezi základní používané informace patří například barva a vzor zobrazený na povrchu. Textury jsou využívány k zobrazování složitých detailů objektů a to bez nutnosti vytváření složité geometrie tělesa. V případě, že zobrazení detailů vyžaduje složitou geometrii objektu, je využití textur ve většině případů efektivnější než samotné vytváření a zobrazování geometrie objektu.

1.1 Typy textur

Existují čtyři používané rozměry textur, kde každý rozměr má jiné využití. Používané jsou jedno, dvou, troj a čtyřrozměrné textury. Jednorozměrné textury mohou být například použity pro určení barevného vzoru křivek (přerušovaný vzor). Trojrozměrné určují hodnotu texelu v prostoru a používají se například při zobrazování různých řezů tělesem. Čtyřrozměrné textury obsahují navíc informace potřebné pro animování trojrozměrných textur. Poslední jsou dvojrozměrné textury. Tyto textury jsou nejčastěji používanou kategorií v počítačové grafice. Jejich využití spočívá v určení vlastností jednotlivých ploch objektu.

Podle toho jakou vlastnost daná textura popisuje, se dají textury rozdělit na několik skupin. Při použití Phongova osvětlovacího modelu je možné rozdělit textury podle vlastností ovlivňujících jednotlivé složky, které určují výsledné osvětlení ve scéně. Difúzní mapy (diffuse map) ovlivňují barvu povrchu, která je dána difúzní složkou. Další vlastností je odraz světla na povrchu tělesa. Tato vlastnost je ovlivněna pomocí zrcadlové složky. Pro změnu této složky existují dva typy textur. První je zrcadlová mapa (specular map) a druhá je mapa odrazů (reflection map) případně environmentální mapa (environment map).

Zrcadlová mapa ovlivňuje koeficient zrcadlového odrazu a tím určuje, které části povrchu mají odrážet světlo a jaká je intenzitu těchto odrazů. Mapa odrazů nebo environmentální mapa umožňuje ovlivnit barvu odrazu povrchu a tím docílit odrážení okolních objektů na lesklém povrchu. Dále lze pomocí textur určit průhlednost povrchu. Tyto textury se nazývají mapy průhlednosti (opacity map). Stejného efektu jako s mapou průhlednosti lze dosáhnout přidáním alfa kanálu do difúzní mapy. Poslední skupinou jsou textury upravující tvar povrchu. Úprava povrchu může být pouze vizuální nebo může dojít ke změně geometrie tělesa. V případě, že jde o změnu pouze vizuální, se jedná o normálové mapy (bump maps). Normálová mapa obsahuje informace o změně normálových vektorů, které jsou následně využívány pro výpočet osvětlení daného povrchu. Při změně geometrie tělesa displacement mapa obsahuje informace o posunu jednotlivých vektorů. V textech [6] a [7] je možné získat více informací o texturách. A text [8] obsahuje podrobnější vysvětlení Phongova osvětlovacího modelu.

1.2 Mapování textur

Mapování textur je proces, při kterém jsou nanášeny textury na povrch objektu. Textury jsou nanášeny na povrch objektu přiřazením texelu pro každý vrchol povrchu. Definování odpovídajících texelů pro všechny body na povrchu je označováno jako inverzní mapování. Souřadnice určující pozici texelu v textuře se označují jako UV souřadnice. Popis mapování textur v této kapitole odpovídá popisu v textu [8].

1.2.1 Inverzní mapování

Při inverzním mapování je potřeba nejdříve pro zobrazovaný pixel na výstupním zařízení určit odpovídající bod plochy v zobrazované scéně. Tento krok je proveden využitím inverzní transformace k transformaci zobrazovací a k transformaci na výstupní zařízení. Zobrazovací transformace je dána maticí realizující středové promítání, která provádí transformaci bodu ze souřadnic scény do souřadnic jednotkového zobrazovacího hranolu. Transformace na výstupní zařízení určuje odpovídající souřadnice na výstupním zařízení pro pozici bodu v jednotkovém zobrazovacím hranolu. Inverzní transformace je aplikována na pixel výstupního zařízení. Po získání bodu ve scéně je následně zjištěna odpovídající pozice v textuře. Získaná hodnota z textury je následně použita při určování hodnoty pixelu na výstupním zařízení.

Většinou jsou objekty tvořeny trojúhelníkovými plochami. Proto je transformace z prostoru scény do prostoru textury popsána pro trojúhelníkovou plochu danou třemi vrcholy V_1, V_2 a V_3. Pozice vrcholů je dána vektorem $V_i(x_i, y_i, z_i)$, kde $i = 1, 2, 3$ je odpovídající označení vrcholu. UV souřadnice texelů odpovídající jednotlivým vrcholům jsou U_1, U_2 a U_3. UV souřadnice $U(u, v)$ jsou většinou normalizovány do intervalu $\langle 0, 1 \rangle$. Při zjišťování odpovídajících texelů je textura umístěna do roviny xy souřadného systému scény. Proto souřadnice texelu (u, v) v textuře odpovídají pozici (x, y) v rovině xy.

Transformaci bodu A z prostoru scény do pozice A_t v prostoru textury je možné zapsat vztahem 1, kde T_1, T_2 a T_3 jsou matice realizující jednotlivé části této transformace. Pozice bodů A a A_t jsou určeny v homogenních souřadnicích.

$$A_t = A \cdot (T_1 T_2 T_3) \tag{1}$$

Matice T_1 provádí posun vrcholu V_1 do počátku souřadného systému scény, který odpovídá v textuře UV souřadnici $(0, 0)$.

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix} \tag{2}$$

Matice T_2 realizuje změnu souřadného systému trojúhelníka na souřadný systém scény. Změna souřadného systému způsobí rotaci všech bodů daného trojúhelníka do roviny xy a umístění bodů V_1 a V_2 na osu x. Pro výpočet rotace je zapotřebí nejdříve zjistit vektory určující osy souřadného systému trojúhelníka. Vektor a určuje směr hrany V_1V_2 a vektor b určuje směr hrany V_1V_3. Kvůli umístění hrany V_1V_2 na osu x souřadného systému scény je vektor a osou x_t souřadného systému trojúhelníka. Pomocí skalárního součinu mezi vektory a,b je zjištěn vektor n určující osu z_t. Poté jsou obě osy normalizovány na jednotkové vektory a skalárním součinem osy x_t a z_t je získána osa y_t.

$$\begin{aligned} a &= V_2 - V_1 \\ b &= V_3 - V_2 \\ n &= a \times b \\ x_t &= \frac{a}{|a|} \\ z_t &= \frac{n}{|n|} \\ y_t &= z_t \times x_t \end{aligned} \tag{3}$$

Označení $|a|$ je velikost vektoru $a(a_1, a_2, a_3)$, která je dána jako $|a| = \sqrt{a_1^2 + a_2^2 + a_3^2}$. Skalární součin dvou třísložkových vektorů je vypočítán následujícím vztahem. [9]

$$x \times y = (x_2y_3 - x_3y_2, x_1y_3 - x_3y_1, x_1y_2 - x_2y_1) \tag{4}$$

Pomocí os $x_t(x_x, y_x, z_x)$, $y_t(x_y, y_y, z_y)$ a $z_t(x_z, y_z, z_z)$ dostaneme matici T_2 v následujícím tvaru.

$$T_2 = \begin{bmatrix} x_x & x_y & x_z & 0 \\ y_x & y_y & y_z & 0 \\ z_x & z_y & z_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Poslední je transformace všech vrcholů trojúhelníka na odpovídající UV souřadnice. Matice realizující tuto transformaci má tvar.

$$T_3 = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ e & f & 0 & 1 \end{bmatrix} \tag{6}$$

Členy v matici T_3 jsou neznámé. Pokud je označena pozice vrcholů po transformacích provedených maticí T_1 a T_2 jako W_i, UV souřadnicím odpovídajícím jednotlivým vrcholům je přidána třetí a čtvrtá složka, takže mají tvar $U(u, v, 0, 1)$, potom je možné jednotlivé členy matice T_3 získat vyřešením soustavy rovnic danou vztahy 7. Pro šest neznámých a, b, c, d, e, f vznikne soustava šesti rovnic.

$$\begin{aligned} U_1 &= W_1 \cdot T_3 \\ U_2 &= W_2 \cdot T_3 \\ U_3 &= W_3 \cdot T_3 \end{aligned} \tag{7}$$

Kvůli normalizaci souřadnic do rozsahu $\langle 0, 1\rangle$ je potřeba po získání pozice $A_t(u, v)$ zjistit, který texel v textuře odpovídá této pozici. Pozice texelu v textuře $T(x, y)$ o rozměrech $M \times N$ je dána vztahy 8.

$$\begin{aligned} x &= u \cdot (M - 1) \\ y &= v \cdot (N - 1) \end{aligned} \tag{8}$$

1.3 Dělení

Dělení textur je možné provádět na základě jejich vzhledu, který je dán například konkrétní barevnou kombinací, vzorem nebo tvarem prvků. Jedno z jednoduchých dělení textur, které alespoň podvědomě zná každý člověk, je označení podle toho jakým způsobem daná textura vznikla. Tato myšlenka umožňuje textury rozdělit do dvou kategorií a to na:

- **přírodní** vyskytující se v přírodě a vytvořené bez zásahu člověka
- **umělé (syntetické)** vymyšlené a vytvořené člověkem.

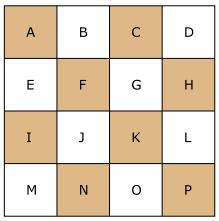
Jako přírodní textury můžeme uvést například mraky na obloze, písek nebo skálu. Mezi umělé textury patří parkety, látka atd.

Další možností jak klasifikovat textury do různých skupin je podle prostorové stejnorodosti neboli homogenity. Prostorová homogenita určuje míru rovnoměrnosti uspořádání prvků a shody vzorů napříč celou texturou. Tato homogenita může být buď lokální, nebo globální. Například na obrázku 1 je šachovnice, kde v každém poli je jiné písmeno. Proto je každé pole lokálně nehomogenní, avšak šachovnice, jako celek zachovává svůj vzor, je globálně homogenní. Podle homogenity rozdělujeme textury na:

- **homogenní** rovnoměrné rozložení prvků podle daného vzoru, kde jednotlivé prvky jsou shodné
- **slabě homogenní** ve vzoru je patrná lokální změna mezi jednotlivými prvky nebo se mění prostorové uspořádání prvků
- **nehomogenní** postrádá opakování vzorů a rozložení prvků není v žádném ohledu možné považovat za podobné.

Nejčastěji používaným způsobem je rozdělení textur do kategorií podle míry náhodnosti, kterou mají jejich vzory a prvky. Zavedením této míry je možné určit dvě základní kategorie, které jsou:

- **regulární** obsahující lehce rozpoznatelné opakující se vzory tvořené malými prvky, které jsou periodicky uspořádány
- **stochastické** náhodné vzory tvořené většinou prvky, které jsou hůře rozpoznatelné.



Obrázek 1: Ukázka homogenity textur.

Všechna dříve popsaná dělení odpovídají dělení v článku [10].

Vzhledem k faktu, že většina textur je kombinací těchto dvou zmíněných kategorií, je toto dělení nedostačující. Zpřesněním tohoto rozdělení je klasifikace textur, která byla použita v článku [11]. Toto rozdělení textur se snaží zpřesnit dříve zmíněné dělení podle míry podobnosti tím, že zároveň dělí textury i podle jejich homogenity. Klasifikace textur do kategorií podle zmíněného článku je následující:

- **strukturované (structured)** pravidelně tvarované prvky, které mají různé velikosti a jsou uspořádány do dlaždicových vzorů
- **regulární (regular)** prvky stejné velikosti a tvaru umístěné do periodicky upořádáného vzoru
- **buňkové (cellular)** prvky různých velikostí i tvarů s nepravidelným uspořádáním, mezi prvky je znatelná spojitost a podobnost ve vzhledu
- **částečně strukturované/stochastické (semi-structured/stochastic)** náhodné uspořádání prvků stejného typu, prvky se mohou lišit tvarem a velikostí
- **velké prvky (large features)** prvky mají náhodný tvar a velikost, jejich upořádání je také náhodné, podobně jako u buňkových textur je i zde patrná podobnost ve vzhledu jednotlivých prvků.

(a) strukturované (b) regulární (c) buňkové (d) stochastické (e) velké prvky

Obrázek 2: Ukázka jednotlivých kategorií pro rozdělení textur (článek [11]).

2 Syntéza textur

Syntéza textur je proces, při kterém jsou generovány nové textury. Textury jsou generovány v závislosti na vzorové textuře. Většinou se jedná o vygenerování nové textury o požadovaném rozlišení ze vzorové textury, která nemá požadovanou velikost a při mapování textury by následně mohlo dojít k viditelnému opakování dané textury na objektu. Nebo v případě špatného napojení dané textury na jejích krajích by vznikly viditelné řezy. Tento problém se snaží vyřešit syntéza textur vygenerováním dostatečně velké textury. Syntézu textur lze popsat jako proces, kdy je na základě vzorové textury vytvářena nová textura o vhodné velikosti tak, aby při pohledu na novou texturu bylo zřejmé, že obě textury zobrazují stejný vzor a aby nebylo znatelné opakování vzorové textury ve vygenerované textuře.

Postup vytvoření textury se skládá ze dvou částí. První částí je analýza, která se zabývá získáním vhodného modelu popisujícího proces vzniku vzorové textury. Získáním takového modelu je možné následně pomocí stejného procesu generovat nové textury. Protože se většina textur pohybuje při jejich klasifikaci mezi strukturovanými a stochastickými texturami, je potřeba, aby model dobře popisoval strukturované i stochastické části textury. Na základě přesnosti vygenerované textury vzhledem k jejímu vzoru je určována kvalita zvoleného modelu. Druhou částí je samotná syntéza, která určuje postup vygenerování nové textury s pomocí vybraného modelu. Podrobnější informace o syntéze textur jsou uvedeny v článcích [1] a [2].

2.1 Markov Random Field

Většina algoritmů pro syntézu textur je založena na modelu nazývaném Markov Random Field (MRF), případně je inspirována tímto modelem. Tento model považuje texturu za náhodné pole, které popisuje hodnotu pixelu v závislosti na pixelech nacházejících se v určeném okolí. Jinak řečeno hodnota každého pixelu je dána hodnotami pixelu v jeho okolí. Využití modelu Markov Random Field umožňuje popsat proces vytváření nové textury jako generování textury tak, aby pro každý pixel vygenerované textury existoval ve vzorové textuře pixel se stejným okolím.

2.2 Metody

V následujících kapitolách jsou popsány základní algoritmy pro syntézu textur. Podle těchto algoritmů jsou metody pro syntézu textur děleny do kategorií, kde názvy jednotlivých kategorií odpovídají názvům použitých algoritmů. Pro zájemce o jiné metody, než ty popsané v následujících kapitolách, je doporučen článek [1], který pokrývá velké množství aktuálních metod pro syntézu textur a slouží jako tutoriál pro zájemce o tuto oblast. Kromě metod generujících 2D textury obsahuje například metody pro syntézu povrchových textur (Surface Texture Synthesis), syntézu textur řízených tokem (Flow-guided Texture Syntheisis), syntézu 3D textur (Solid Texture syntesis) atd.

2.2.1 Pixel-based

Základem těchto algoritmů je kopírování pixelů ze vzorové textury do syntetizované textury. Pixely jsou při vytváření textury kopírovány po jednom. Výběr pixelu, který bude nakopírován, probíhá následovně. Z výsledné textury je vybrána pozice, do které ještě nebyl nakopírován žádný pixel. Pro danou pozici jsou zjištěny všechny již nakopírované pixely nacházející se v jeho sousedství. Velikost okolí je specifikována uživatelem. Na základě hodnot sousedních pixelů jsou vybrány všechny pixely ve vzorové textuře, tak aby tyto hodnoty co nejblíže odpovídaly hodnotám zjištěným pro okolí aktuálně zpracovávaného pixelu. Poté je z množiny těchto pixelů vybrán jeden náhodně. Protože je při hledání vhodného pixelu potřeba porovnávat pixely v jeho okolí, je prvním krokem inicializace textury nakopírováním náhodné malé části vstupní textury do výsledné textury. Nové pixely jsou poté přidávány kolem této části, dokud není zaplněna celá textura. Tento algoritmus je popsán v článku Texture synthesis by non-parametric sampling [12].

Všechny metody z této kategorie zvětšují texturu po jednom pixelu, ale liší se ve způsobu výběru jednotlivých pixelů. Například Fast texture synthesis using tree-structured vector quantization [13] využívá neměnného počtu sousedních pixelů a pixely jsou přidávány do výsledné textury v pořadí rastrového průchodu. Inicializace probíhá náhodným kopírováním pixelů do výsledné textury. Pevně daný počet sousedních pixelů umožňuje zrychlení syntézy použitím algoritmů využívajících pro výpočet stromové struktury.

2.2.2 Patch-based

Algoritmy spadající do této kategorie se snaží vylepšit výsledky syntézy tím, že místo kopírování jednoho pixelu do výsledné textury kopírují celé bloky sousedních pixelů. Tyto bloky bývají nazývány patche. Kategorie patch-based metod je z jistého pohledu podobná kategorii pixel-based. Taky vybírá jednotlivé patche tak, aby co nejlépe odpovídaly již nakopírovaným patchům.

Nejdříve je provedena inicializace textury. Ve většině algoritmů jde o nakopírování náhodného bloku ze vzorové textury do syntetizované textury na předem určené místo. Stejně jako je u pixel-based algoritmů porovnáváno okolí aktuálně zpracovávaného pixelu, je i u patch-based algortimů prováděno podobné porovnávání. Část nového patche se překrývá s již nakopírovanými patchi, aby bylo možné porovnávat kandidáty na nové patche s již nakopírovanými patchi. Na základě překrytí patchů je následně určeno, zda se daný patch hodí na danou pozici do syntetizované textury nebo ne. Pro výběr pozice nového patche je využíváno podobných principů jako u pixel-based algoritmů. Prvním je pořadí rastrového průchodu a druhým způsobem je postupné rozšiřování textury přidáváním patchů k prvnímu nakopírovanému patchi.

Největším rozdílem mezi jednotlivými algoritmy využívajícími kopírování po patchích je způsob, kterým se vypořádávají s překrývajicími oblastmi patchů. Například Real-time texture synthesis using patch-based sampling [14] používá míchání barev (blending) těchto oblastí. Lapped textures [15] kopíruje patche nepravidelných tvarů a novými patchi přepisuje ty staré.

Jedním z možných řešení těchto oblastí je nalezení řezu, který určuje nejlepší napojení patchů. Metody v článcích Image Quilting for Texture Synthesis and Transfer [3] a Graphcut Textures: Image and Video Synthesis Using Graph Cuts [4] využívají právě takového přístupu. Tyto dvě právě zmíněné metody byly vybrány pro implementaci a jejich následné srovnání v této práci. Metoda Image Quilting byla vybrána z důvodu jejího výskytu ve většině publikací zabývajícími se syntézou textur. Druhá metoda byla vybrána z důvodu, že také zjišťuje řez mezi patchi. Avšak využívá jiného přístupu pro umisťování patchů a také jiného algoritmu pro zjištění optimálního řezu. Podrobný popis metody Image Quilting je v kapitole 3. Bližší informace k metodě Graphcuts je možné nalézt v kapitole 4.

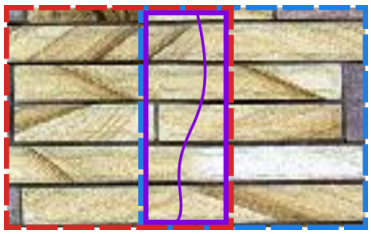
2.2.3 Texture optimization

Metody v této kategorii využívají kombinace přístupů použitých v pixel-based a patch-based metodách. Z kategorie pixel-based využívají přístup vytváření textury po jednom pixelu. Avšak neprovádějí pouze kopírování již existujících pixelů ze vzorové textury. K určení hodnoty aktuálně zpracovávaného pixelu využívají funkci energie (energy function). Funkce je dána rozdílem hodnot sousedů ve vzorové textuře a v syntetizované textuře umocněných na druhou. Výsledná hodnota pixelu je určena pomocí minimalizace této funkce. Minimalizace funkce probíhá ve dvou krocích. V prvním kroku je metodou nejmenších čtverců určena hodnota pixelu ve výsledné textuře pomocí vybraného sousedství ve vzorové textuře. V druhém kroku je pro sousedství ve výsledné textuře nalezeno odpovídající sousedství ve vzorové textuře. Tyto dva kroky jsou opakovány, tak dlouho dokud nedojde ke konvergenci nebo neproběhne maximální počet iterací. Pro bližší informace k této kategorii je doporučen článek [16].

3 Image Quilting

Image Quilting je metodou umožňující generování textur. Naimplementovaný algoritmus pro účely této práce odpovídá algoritmu v článku [3]. V publikaci je prezentováno také využití daného algoritmu pro přenos textur. V následujících kapitolách je popsána metoda pouze z pohledu syntézy textur. V případě zájmu o přenos textur je možné najít další informace ve zmíněném článku.

Metoda patří do kategorie patch-based, která je založena na kopírování částí ze vzorové textury do textury o větším rozlišení. Tyto části jsou bloky textury a jsou nazývány patche. Nejdříve je potřeba zvolit vstupní texturu, jejíž rozlišení je potřeba zvětšit, a určit vstupní parametry. Poté jsou postupně po jednom kopírovány jednotlivé bloky do výstupní textury o zvoleném rozlišení. Bloky jsou kopírovány na předem určené pozice. Zvolení konkrétní části, která má být z vstupní textury zkopírována na danou pozici, probíhá na základě porovnání podobnosti mezi aktuálně syntetizovanou texturou a vstupní texturou. Při umísťování bloků nedochází pouze k jejich skládání vedle sebe, ale bloky umístěné vedle sebe se překrývají (obrázek 5). Právě tato překrývající se oblast je použita k určení míry podobnosti bloků. Ze všech bloků jsou vybrány ty, které splňují předem zvolené omezení pro podobnost. Po vybrání vhodného bloku na zkopírování do výstupní textury je zapotřebí v překrývající se oblasti zjistit nejlepší napojení mezi těmito bloky. Toto napojení je určeno na základě cesty procházejícího přes oblast překrytí. Cesta je určena tak, aby rozdíl mezi bloky v pixelech této cesty byl minimální (obrázek 3). Provedením těchto kroků pro všechny pozice nových bloků je získána syntetizovaná textura.

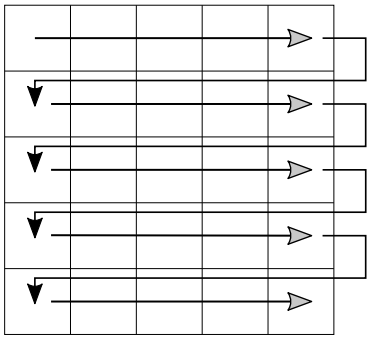


Obrázek 3: Určení cesty pro napojení bloků.

Všechny parametry volené na začátku algoritmu a detaily k jednotlivým krokům algoritmu jsou popsány v následujících kapitolách.

3.1 Algoritmus

Vstupem do algoritmu je vzorová textura. Tento vstupní obraz je značen jako I. Výstupní obraz O je potom syntetizovaná textura, která je výsledkem syntézy. Hodnota pixelu na pozici (x, y) v obraze I je značena $I(x, y)$. Nejprve je potřeba upřesnit v jakém pořadí jsou procházeny pozice jednotlivých bloků. Jinak řečeno určit na které místo v obraze O je zkopírován aktuálně hledaný blok a s kterými bloky se překrývá. Pozice bloku je dána umístěním levého horního rohu v obraze O. K vytváření obrazu O je použit rastrový průchod. Obrázek 4 je grafickým znázorněním takového průchodu. Pro zjednodušení se jednotlivé bloky v obrázku nepřekrývají.



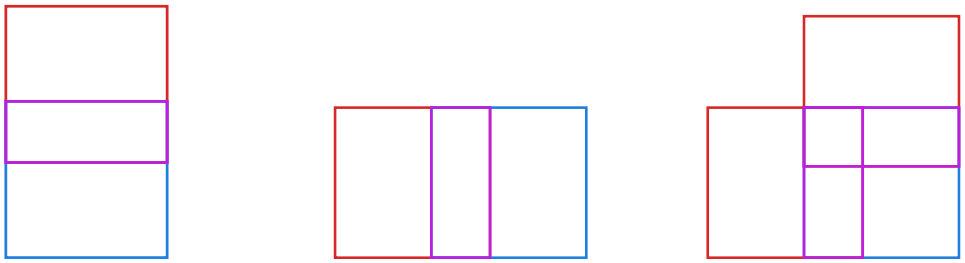
Obrázek 4: Rastrový průchod syntetizovaného obrazu O po jednotlivých blocích.

Pro určení pozic je zapotřebí znát velikost oblasti, ve které se sousední bloky překrývají. Označením velikosti tohoto překrytí jako V a pokud K je šířka a L je výška bloku, potom je možné určit horizontální vzdálenost bloků $dist_x = K - V$ a vertikální vzdálenost $dist_y = L - V$.

První blok je vždy zkopírován do levého horního rohu obrazu O. Pro tento blok nedochází k porovnávání podobnosti. Z vstupní textury je pouze náhodně vybrána oblast o velikosti $K \times L$. Všechny následující bloky jsou voleny podle míry podobnosti.

3.1.1 Výpočet podobnosti bloků

Výpočet podobnosti bloků probíhá v oblastech, kde se překrývá místo vyhrazené pro aktuálně zpracovávaný blok s již zpracovanými bloky. Při rastrovém průchodu dochází k třem různým způsobům překrytí (obrázek 5). Pro celý první řádek dochází pouze k vertikálnímu překrytí a naopak pro celý první sloupec pouze k horizontálnímu. U ostatních pozic bloků se vytváří překrytí mající obrácený tvar písmene L.



(a) vertikální (b) horizontální (c) tvar písmena L

Obrázek 5: Typy překrytí vznikající při syntéze. Červeně jsou znázorněny bloky, které již byly nakopírovány do výsledného obrazu. Modře je označena pozice aktuálně zpracovávaného bloku. Fialová určuje oblast překrytí.

Podobnost dvou bloků je určena chybou v oblasti překrytí. K výpočtu této chyby je použita L_2 norma nad hodnotami pixelů daná vzorcem 9. Tato norma je označována pomocí $\|\cdot\|$. Kde x je vektor o třech složkách (r, g, b). [17]

$$\|x\| = \sqrt{r^2 + g^2 + b^2} \tag{9}$$

Velikost bloku je $K \times L$ a velikost překrytí je V. Pokud je posun na pozici zpracovávaného bloku v obraze O označen jako t a posun v obraze I jako s, kde t a s jsou vektory určující posun v ose x a v ose y, je možné napsat výpočet chyby dvou bloků E vzorcem 10. Vzhledem k faktu, že dochází k různým typům překrytí je vzorec 10 platný pouze pro horizontální překrytí. Vzorec 11 udává chybu pro vertikální překrytí. Chyba pro překrytí tvaru L je vypočtena pomocí vztahu 12, kde $E_{horizontal}$ a $E_{vertical}$ značí výsledky vzorců 10 a 11.

$$E_{horizontal} = \sum_{x=0}^{K} \sum_{y=0}^{V} \|(O(x + t_x, y + t_y) - I(x + s_x, y + s_y))\| \tag{10}$$

$$E_{vertical} = \sum_{x=0}^{V} \sum_{y=0}^{L} \|(O(x + t_x, y + t_y) - I(x + s_x, y + s_y))\| \tag{11}$$

$$E_{Lshape} = E_{horizontal} + E_{vertical} - E_{diffence} \tag{12}$$

$E_{difference}$ ve vzorci 12 je oblast, která byla do výsledné chyby započítána dvakrát. Výpočet chyby pro tuto oblast je možné zapsat jako vztah 13.

$$E_{difference} = \sum_{x=0}^{V} \sum_{y=0}^{V} \|(O(x + t_x, y + t_y) - I(x + s_x, y + s_y))\| \tag{13}$$

Pro každou pozici zpracovávaného bloku jsou vypočteny chyby překrytí pro všechny posuny, tak aby velikost bloku při posunu s byla $K \times L$. Z těchto posunů jsou vybrány ty, jejichž chyba E splňuje podmínku danou vztahem 14. Kde k je parametr určující míru náhodnosti výběru bloku. Následně je ze všech posunů s, které splňují podmínku, vybrán náhodně jeden. Nad tímto blokem je proveden výpočet minimální cesty.

$$E < E_{min} + E_{min} \cdot k \tag{14}$$

3.1.2 Výpočet minimální cesty

K určení cesty je zapotřebí zjistit povrchovou chybu mezi oblastmi překrytí. Vypočtením této chyby, kterou označíme jako E_{surf} vznikne matice, kde $E_{surf}(x, y)$ udává chybu na pozici (x, y). Podle vzorce 15 je proveden výpočet chyby na pozici (x, y).

$$E_{surf}(x, y) = (O(x + t_x, y + t_y) - I(x + s_x, y + s_y))^2 \tag{15}$$

Po vypočtení chyby E_{surf} je možné přejít k dalšímu kroku pro určení cesty. Dalším krokem je na základě E_{surf} vytvořit kumulativní chybový povrch. Kumulativní chybový povrch je značen jako E_c. $E_c(x, y)$ potom obsahuje kumulativní chybu na pozici daného pixelu, která je vypočtena pomocí vzorce 16 pro vertikální cestu a vzorce 17 pro horizontální cestu.

$$E_c(x, y) = \begin{cases} E_{surf}(x, y) & y = 0 \\ E_{surf}(x, y) + min(E_c(x + 1, y - 1), E_c(x, y - 1), E_c(x - 1, y - 1)) & y > 0 \end{cases} \tag{16}$$

$$E_c(x, y) = \begin{cases} E_{surf}(x, y) & x = 0 \\ E_{surf}(x, y) + min(E_c(x - 1, y + 1), E_c(x - 1, y), E_c(x - 1, y - 1)) & x > 0 \end{cases} \tag{17}$$

Jakmile je vytvořena kumulativní chyba pro všechny pixely v překrývající se oblasti je možné nalézt minimální cestu. Nejmenší hodnota v posledním řádku (sloupci) E_c pro vertikální (horizontální) cestu udává pozici pixelu ukončujícího cestu. Vzhledem k způsobu jakým je kumulativní chyba počítána je možné z pixelu ukončujícího cestu zpětně stopovat cestu až k jejímu začátku. Na obrázku 6 je příklad výpočtu minimální cesty v obraze o rozměrech 5×4 pixely.

4	1	3	8
6	7	5	4
4	8	2	6
7	3	5	9
9	1	6	2

(a) Povrchová chyba.

4	1	3	8
7	8	6	7
11	14	8	12
18	11	13	17
20	12	17	15

(b) Kumulativní chyba.

4	**1**	3	8
6	7	**5**	4
4	8	**2**	6
7	**3**	5	9
9	**1**	6	2

(c) Minimální cesta.

Obrázek 6: Výpočet minimální vertikální cesty pro překrývající se oblasti. Obrázek (a) znázorňuje vypočtenou povrchovou chybu E_{surf} a obrázek (b) kumulativní povrchovou chybu E_c vypočtenou z obrazu (a). Na obrázku (c) jsou modře znázorněny pixely tvořící minimální vertikální cestu obrazem (a).

V případě, že je nutné vypočítat minimální cestu pro blok s horizontálním i vertikálním překrytím (obrázek 5c), jsou nezávisle na sobě vypočítány obě cesty. Po vypočtení obou cest, jsou nalezeny všechny průsečíky horizontální a vertikální cesty. Poté je vybrán průsečík, který má nejmenší hodnotu kumulativní chyby. Tento průsečík se stává novým společným počátečním pixelem pro obě cesty.

3.2 Parametry

V této kapitole je popsán vliv všech parametrů algoritmu na výslednou syntézu textury. Všechny zde uvedené parametry jsou volitelné uživatelem. Mezi volitelné parametry patří velikost výsledného obrazu O, velikost bloku $K \times L$, velikost překrytí sousedních bloků V a parametr k určující míru náhodnosti vybíraného bloku.

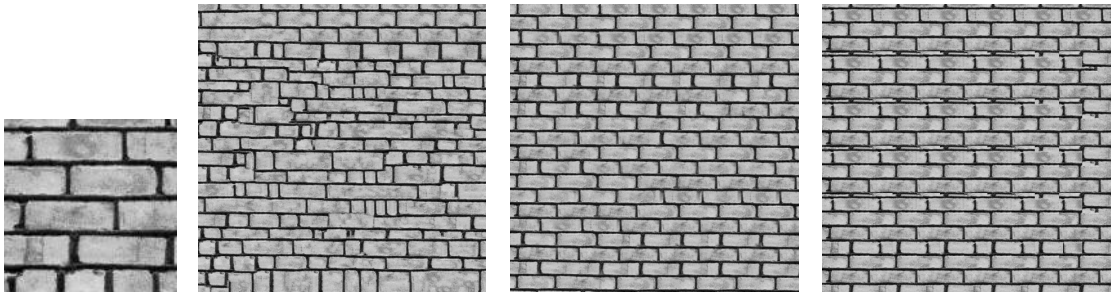
Pokud je ve vzorové textuře nějaký objekt, který je výrazný nebo neobvyklý, potom čím větší je syntetizována textura, tím větší je šance, že se tento objekt bude opakovat. Při opakování takového objektu dochází k vytváření dojmu, že syntetizovaná textura není reálná. Dalším problémem při vytváření textur o rozměrech několikanásobně větších než rozměry vzorové textury, je vznikání vzorů, které ve vzorové textuře nejsou. Tyto vzory většinou vznikají z důvodu nedostatku informací ve vzorové textuře. Nedostatkem informací je zde myšlena malá různorodost textury nebo pro některé bloky textury chybějící vhodné bloky k navázání.

Posledním a největším problémem je navyšující se pravděpodobnost výskytu tzv. artefaktů. Artefakt je označení pro místo v textuře, kde se setkávají dva bloky a v kterém došlo k výběru bloku špatně navazujícího na ostatní bloky. Čím větší je rozdíl rozměrů mezi vzorovou a syntetizovanou texturou tím větší je šance na výskyt artefaktů.

(a) vzor (b) 200x200 (c) 300x300

Obrázek 7: Vliv velikosti syntetizované textury.

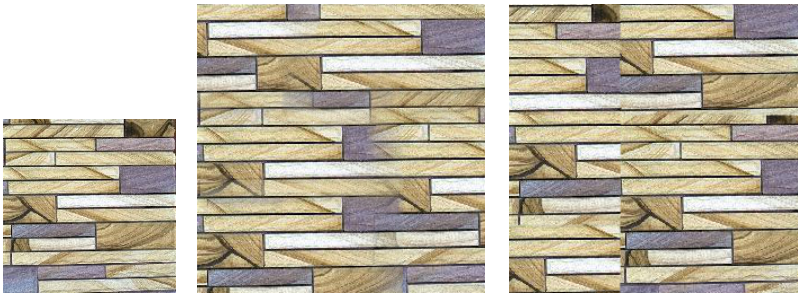
Velikost bloků je nutné volit tak, aby zachytila vzor textury. Na obrázku 8b je vidět zvolení příliš malé velikosti bloků. Při syntetizování nebyl zachycen vzor a vzniklá textura obsahuje množství oblastí, které neodpovídají vzorové textuře. Naopak obrázek 8d ukazuje výsledek syntézy pro velikost bloků blízkou velikosti vzorové textury. Takto zvolená velikost bloků omezuje počet možností pro posun s. To má za následek, že vzniklá textura je pouze nakopírováním špatně se překrývajících bloků vedle sebe.



(a) vzor (b) 15x15 (c) 60x60 (d) 120x120

Obrázek 8: Vliv velikosti bloku na syntetizovanou texturu. Obrázek (c) ukazuje výsledek syntézy pro velikost bloků zvolenou přibližně mezi extrémními hodnotami (b) a (d).

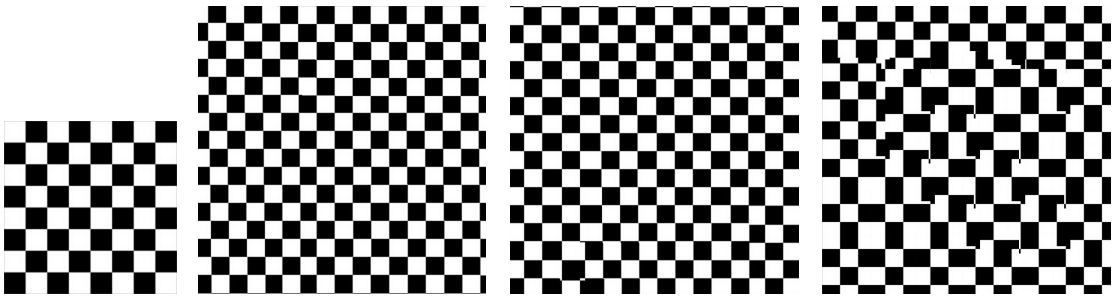
Na základě oblasti kde se bloky překrývají, je určována podobnost bloků. A také je v této oblasti prováděn výpočet minimální cesty. Malá velikost překrytí způsobuje nedostatek informací pro správné navázání bloků. Obrázek 9c je výsledkem syntézy, kde velikosti překrytí byla zvolena jako třicetina velikosti bloků. Na tomto obrázku je možné v pravém dolním rohu vidět špatné navázání bloků způsobené nedostatečnou velikostí překrytí. Na obrázku 9b je ukázka syntézy pro velikost překrytí rovnu třetině velikosti bloků.



(a) vzor (b) $V = K/3$ (c) $V = K/30$

Obrázek 9: Vliv velikosti překrytí bloků V na syntetizovanou texturu. Velikost bloku je $K \times L$, kde $K = L$.

Parametr k ze vztahu 14 má vliv na omezení výběru bloků, které jsou uvažované pro zkopírování do výsledné textury. Z podmínky vyplývá, že čím větší je hodnota k, tím náhodnější je výběr bloků. A naopak čím menší je hodnota k, tím více je výběr striktní.



(a) vzor (b) $k = 0.1$ (c) $k = 1$ (d) $k = 5$

Obrázek 10: Vliv parametru k ze vztahu 14 na syntetizovanou texturu. Parametr určuje míru náhodnosti pro výběr bloků.

3.3 Využití detekce hran

V případě některých textur je důležitější zachování vzoru a správného navázaní bloků i za cenu nevyužití informace o barvě pixelů pro porovnávání podobnosti. Proto v souvislosti s algoritmem Image Quilting bylo provedeno testování využití detekce hran pro zlepšení výsledků syntézy textur. V následujícím textu jsou uvedeny a popsány testované možnosti pro detekci hran. Všechny uvedené detekce hran jsou zpřístupněny uživateli k použití. Popis detekcí hran v této kapitole odpovídá popisu v článcích [5] a [18].

Detekce hran je aplikována na vstupní obraz I. Provedením detekce hran nad I dostaneme obraz S obsahující informace o hranách v obraze. Obraz S má stejné rozměry jako I a na pozici $S(x, y)$ obsahuje hodnotu udávající velikost hrany na pozici pixelu (x, y). Výsledný obraz S je dále použit pro určení podobnosti bloků a výběr nejvhodnějšího bloku.

A také slouží pro výpočet minimální cesty v překrývající se oblasti. Pro testování byly použity tři metody umožňující zjištění hran v obraze a to Sobelův operátor, Laplaceův operátor a Cannyho detektor hran.

Všechny uvedené metody určují velikost hrany výpočtem gradientu obrazové funkce. K určení velikosti hrany je používána konvoluce obrazové funkce s konvoluční maskou (obrázky 11 a 12). Konvoluční maska je značena jako G a pozice v masce je $G(x, y)$. Konvoluce mezi obrazem I a maskou G v bodě (x, y) je potom značena $(I * G)(x, y)$. Kde velikost masky G je $M \times N$. Indexování v masce G je změněno z $x = 0, \ldots, M-1$; $y = 0, \ldots, N-1$ na $x = -P, \ldots, P$; $y = -Q, \ldots, Q$. Pro $P = (M-1)/2$ a $Q = (N-1)/2$. Za předpokladu lichých hodnot M a N. Potom vztah 18 je zápisem konvoluce masky G s obrazem I.

$$(G * I)(x, y) = \sum_{m=-P}^{P} \sum_{n=-Q}^{Q} G(m, n) \cdot I(x + m, y + n) \tag{18}$$

Pro vypočtení Sobelova operátoru je zapotřebí provést zvlášť pro všechny body v obraze konvoluci s konvoluční maskou provádějící výpočet derivace ve směru osy x (obrázek 11a) a konvoluci s konvoluční maskou provádějící výpočet derivace ve směru osy y (obrázek 11b). Poté můžeme velikost hrany v bodě (x, y) určit vztahem19, kde G_x a G_y jsou derivace ve směru osy x a y. Pro výpočet je možné využít konvoluční masky o větším rozměru než ty, které jsou ukázány na obrázku 11. [19]

$$S(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2} \tag{19}$$

-1	0	1
-2	0	2
-1	0	1

(a) Ve směru osy x.

-1	-2	-1
0	0	0
1	2	1

(b) Ve směru osy y.

Obrázek 11: Konvoluční masky pro výpočet Sobelova operátoru o rozměru 3×3. Maska (a) slouží k výpočtu Sobelova operátoru ve směru osy x a (b) ve směru osy y.

K výpočtu Laplaceova operátoru je pro všechny body obrazu provedena konvoluce obrazu I s konvolučními maskami (obrázek 12). Maska na obrázku 12a počítá druhé derivace ve směru osy x a zároveň směru osy y. Laplaceův operátor má jiné masky pro výpočet derivací bodů, které jsou na krajích nebo v rozích obrazu. Obrázek 12b je ukázkou masky pro body v posledním řádku a obrázek 12c pro body v pravém spodním rohu. Masky pro ostatní speciální případy jsou získány otočením těchto dvou masek.

0	1	0
1	-4	1
0	1	0

(a) Uvniř obrazu.

0	1	0
1	-3	1
0	0	0

(b) Okraj obrazu.

0	1	0
1	-2	0
0	0	0

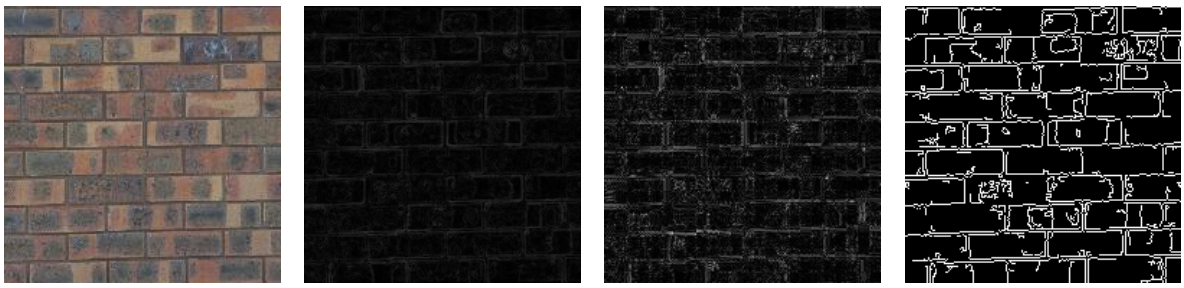
(c) Roh obrazu.

Obrázek 12: Obrázek (a) je konvoluční maska pro výpočet Laplaceova operátoru ve směrech os x a y. Maska (b) slouží k výpočtu na krajích obrazu a (c) v rozích obrazu.

Cannyho detektor hran využívá sofistikovanější přístup pro detekci hran než předchozí zmíněné operátory. Nejdříve je na obraz I použita filtrace pomocí gaussiánu. Tato filtrace snižuje šanci na chybnou detekci hrany. Následně jsou vypočteny derivace G_x ve směru osy x a G_y ve směru osy y. Poté je zjištěna velikost hrany vztahem 19. Jednou z podmínek detektoru je, aby vzdálenost mezi detekovanými pixely hrany a opravdovými pixely hrany byla co nejmenší. Proto je v bodě detekovaná hrana pouze tehdy, je-li v daném bodě lokální maximum. Poslední podmínkou je, že velikost hrany v daném bodě by měla mít dostatečnou velikost. K určení, zda je velikost hrany dostačující, je použito prahování s hysterzí.

Při prahování s hysterzí jsou nejdříve specifikovány hodnoty prahu a to t_{lower} a t_{upper}. Všechny pixely mající velikost hrany větší než t_{upper} jsou označeny za hrany. Pokud je velikost hrany menší než t_{lower} je hrana zamítnuta. Když je velikost hrany mezi hodnotami, je bod označen za hranu právě tehdy, je-li spojen s bodem, jehož velikost hrany je větší než t_{upper}. [20]

V případě zájmu o bližší informace týkající se detekce hran je doporučen text [5].



(a) (b) (c) (d)

Obrázek 13: Ukázka výsledků detekce hran. (a) je obraz nad kterým byla provedena detekce hran. Obrázek (b) je výsledkem Sobelova operátoru, (c) je výsledkem Laplaceova operátoru a (d) je ukázkou výsledku Cannyho detektoru hran.

Jak je patrné například z obrázku 13c, metody na detekci hran jsou náchylné na šum, který má za následek detekování falešných hran v obraze. Tento problém bývá zpravidla řešen filtrací šumu před samotnou detekcí hran. V aplikaci je umožněna filtrace obrazu gaussiánem a mediánem. Filtrace gaussiánem je prováděna konvolucí obrazu s konvoluční maskou pro gaussián. Obrázek 14 ukazuje příklad takové masky. Při filtraci mediánem je každý pixel nahrazen mediánem svých sousedních pixelů v čtvercové oblasti kolem daného pixelu. [21]

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Obrázek 14: Konvoluční maska pro filtraci gaussiánem. Výsledek konvoluce je vydělen součtem čísel v konvoluční masce.

Pro implementaci detekcí hran v obraze a filtrací obrazu, které byly popsány v této kapitole, byla použita volně dostupná knihovna OpenCV.

3.4 Míchání barev v oblasti řezu

I přes výpočet minimální cesty při navazování bloků je v mnoha případech řez mezi bloky znatelný. Problém je možné řešit mícháním barev (blending) v oblasti, kde je řez mezi bloky umístěn. Tento postup byl zvolen i pro úpravu syntetizovaných textur v této práci. Podle vztahu 23 je určena výsledná barva v obrazu $O(x, y)$, kde α je koeficient určující vzdálenost od místa řezu. Výpočet této vzdálenosti je rozdílný pro vzdálenost za řezem a před řezem (vztah 20). Pozice řezu je označená jako p. V je velikost překrytí bloků. Hodnota $step_b$ udává vzdálenost dvou bodů před řezem a $step_a$ vzdálenost za řezem.

0	0.1	0.2	0.3	0.4	p	0.75	1

Obrázek 15: Ukázka míchání barev pro vertikální řez. Modře je vyznačen bod p, kterým prochází řez. $step_b = 0.1$, $step_a = 0.25$. V jednotlivých bodech jsou uvedeny hodnoty α.

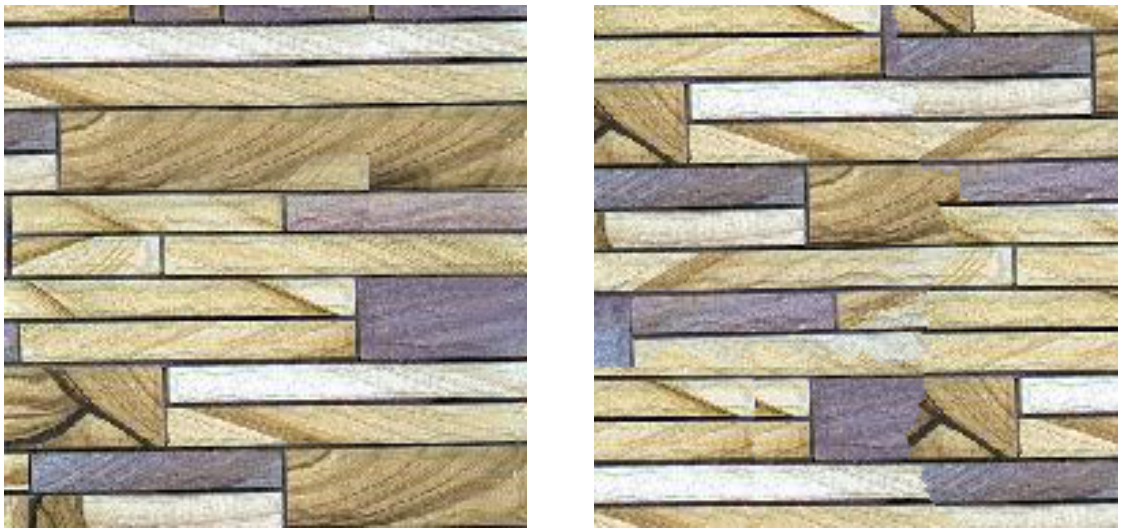
$$\alpha = \begin{cases} x \cdot step_b & x \leq p \\ (x - p) \cdot step_a + 0.5 & x > p \end{cases} \tag{20}$$

$$step_b = \frac{0.5}{p} \tag{21}$$

$$step_a = \frac{0.5}{(V - p - 1)} \tag{22}$$

$$O(x, y) = (1 - \alpha) \cdot O(x, y) + \alpha \cdot I(x, y) \tag{23}$$

Uvedené vztahy platí pro vertikální řez. Vztahy pro horizontální řez lze jednoduše z těchto vztahů odvodit. V případě překrytí tvaru L je provedeno zvlášť míchání horizontálního a vertikálního řezu. V místě, kde dochází k oběma řezům zároveň, je pro bod (x, y) určena barva kombinací obou míchání a to tak, že horizontální a vertikální míchání má na daný bod poloviční vliv.



(a) blending (b) bez blendingu

Obrázek 16: Ukázka vlivu míchání barev na výsledek syntézy.

4 Graphcuts

Metoda byla publikována v článku Graphcut Textures: Image and Video Synthesis Using Graph Cuts [4]. Stejně jako metoda popsaná v kapitole 3 i tato metoda patří do kategorie patch-based. V článku je popsána kromě syntézy textur z jednoho obrazu také syntéza videa. Jelikož se tato práce zabývá pouze syntézou textur z jednoho obrazu, jsou zájemci o syntézu videa nebo o využití algoritmu pro aplikace umožňující interaktivní spojování a míchání více obrazů odkázáni na již zmiňovaný článek.

Ve své práci autoři uvádějí několik různých verzí algoritmu a to konkrétně verze nazvané náhodné umisťování, porovnávání celých bloků a porovnávání částí bloků. Naimplementovaný a dále popsaný algoritmus odpovídá jejich verzi nazvané porovnávání celých bloků.

Základem této metody je vytváření syntetizované textury kopírováním nepravidelně tvarovaných bloků ze vzorové textury do výsledného obrazu. Jakmile je zvolena vstupní textura, která je použita k vytvoření výsledného obrazu, je nejdříve provedeno nakopírování prvního bloku. Po nakopírování prvního bloku jsou následující bloky vybírány tak, aby jejich umístění ve výsledné textuře bylo co nejlepší. To zda je umístění bloku na danou pozici vyhovující, je určeno výpočtem ceny určující podobnost bloků. Tato podobnost je určována pouze v oblasti, kde se kandidát na nově umístěný blok překrývá s bloky již umístěnými ve výsledné textuře. Následuje výpočet pravděpodobnosti výběru daného bloku pro všechny potencionální umístění nových bloků. Na základě této pravděpodobnosti je náhodně vybráno umístění nového bloku. V této fázi algoritmu ještě nedochází k nakopírování bloku do výsledné textury, ale je pro překrývající se části nového bloku a již vybraných bloků vytvořena reprezentace grafem.

Graf je použit k výpočtu minimálního řezu. Tento řez slouží k zjištění nejlepšího napojení mezi bloky a určuje, které pixely v překrývající se oblasti mají být nakopírovány z nového bloku a které mají zůstat z bloků již nakopírovaných. Při vytváření grafu jsou do něj zakomponovány také všechny předchozí vytvořené řezy. Využití předchozích řezů umožňuje jejich vylepšení nahrazením novým řezem nebo případným úplným odstraněním. Algoritmus končí ve chvíli, kdy je do každého bodu výsledné textury nakopírován nějaký pixel ze vzorové textury.

Podrobnosti k jednotlivým krokům algoritmu a také k implementaci jsou popsány v následujících kapitolách.

4.1 Algoritmus

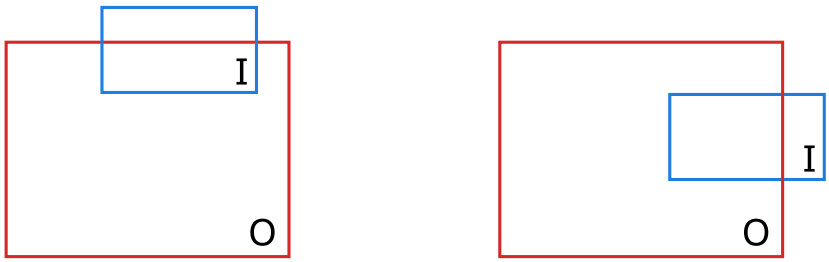
Vstupem do algoritmu je vzorová textura neboli vstupní obraz, který je značen $I(x, y)$. Výsledkem algoritmu je syntetizovaná textura. Tento výstupní obraz je označen $O(x, y)$. Velikost obrazu O je $M \times N$. Hodnota pixelu na pozici (x,y) v obraze je například pro obraz I značena $I(x, y)$. Pokud je v následujícím textu zmíněna pozice bloku, je touto pozicí myšlena pozice (x, y) levého horního rohu bloku vzhledem k obrazu $O(x, y)$.

Jako první je do výsledné textury O umístěn první blok. Prvním blokem je celý vstupní obraz I a je nakopírován na pozici $O(x, y)$, kde $x = 0$ a $y = 0$. Při výběru následujících bloků je nejprve určeno, zda je aktuálně zpracovávaná pozice bloku vhodná. A pouze v případě vhodné pozice je zjišťována cena nakopírování tohoto bloku.

4.1.1 Výběr vhodných pozic

Pozice je určena posunem t obrazu I v obraze O. Pro výběr vhodných pozic je využito dvou pomocných obrazů. A to obrazů S a T. Obraz S slouží jako maska určující zda pixel na pozici (x, y) v obraze O již byl nahrazen novou hodnotou. Pokud byl na pozici (x, y) již zkopírován nový pixel, tak $S(x, y) = 1$ jinak $S(x, y) = 0$. Obraz T je integrálním obrazem obrazu S. Velikost obou obrazů je $M \times N$, která je shodná s velikostí obrazu O. Velikost obrazu I je $K \times L$.

Při vybírání pozice dochází k několika speciálním případům, kdy se nepřekrývá celá oblast obrazu I s obrazem O. Dva příklady těchto případů jsou ukázány na obrázku 17. Jak je patrné z obrázku, bude potřeba určit rozsah hodnot pro posun t. Posun t může nabývat hodnot $t_x = -(K-1), \ldots, M-1$ a $t_y = -(L-1), \ldots, N-1$.



(a) (b)

Obrázek 17: Ukázka dvou speciálních případů vznikajících při výběru pozice nového bloku.

Pro tyto speciální případy je potřeba definovat výšku a šířku oblasti, která je při daném posunu t uvnitř oblasti obrazu O. Výška oblasti je značena H a šířka oblasti W. Potom rozměry oblasti jsou $W \times H$. Způsob definování rozměrů oblasti popisují vztahy 24 a 25.

$$W(t_x) = \begin{cases} K & t_x \geq 0 \wedge t_x \leq (M-K) \\ K + t_x & t_x < 0 \\ K - t_x & t_x > (M-K) \end{cases} \tag{24}$$

$$H(t_y) = \begin{cases} L & t_y \geq 0 \wedge t_y \leq (N-L) \\ L + t_y & t_y < 0 \\ L - t_y & t_y > (N-L) \end{cases} \tag{25}$$

Výběr pozice je omezen podmínkou, že součet hodnot v obraze S v překrývající se oblasti musí být menší než $(W \cdot H) - J$ a zároveň větší než J, kde $J = (W \cdot H)/6$. Tento součet hodnot udává velikost plochy, která je již zaplněna novou texturou. Podmínkou je omezen výběr pozic pouze na ty, které o nějakou část zvětší aktuálně syntetizovanou texturu a zároveň se překrývají s již vybranými bloky. A také je tím zamezeno zvolení již vybrané pozice.

$$J < A(t_x, t_y) < (W \cdot H) - J \tag{26}$$

K výpočtu velikosti plochy je využit integrální obraz T, jehož ukázka je na obrázku 18. Pokud je velikost plochy označena A je možné podmínku zapsat vztahem 26 a velikost plochy vypočítat vztahem 29. Kde $B = T(t_x - 1, t_y - 1)$, $C = T(t_x + W_c, t_y + H_c)$, $D = T(t_x - 1, t_y + H_c)$ a $E = T(t_x + W_c, t_y - 1)$ a W_c, H_c jsou dány vztahy 27, 28. Při výpočtu velikosti plochy může dojít k situaci, kdy budou pro obraz T vyžadovány hodnoty mimo rozsah $0, 1, \ldots, M-1$ a $0, 1, \ldots, N-1$. Proto je nadefinován obraz T mimo tento rozsah a to tak, že hodnota T vně uvedeného rozsahu je rovna 0. [22]

1	1	1
1	1	0
0	0	0

(a) Maska S.

1	2	3
2	4	5
2	4	5

(b) Integrální obraz T.

Obrázek 18: Ukázka masky S (a) a integrálního obrazu T masky S (b) o velikosti 3×3 pixely. Zeleně je označena oblast, ve které je zjišťována hodnota A. Podle vztahu pro výpočet $A(x, y)$ je $A(1, 1) = 5 + 1 - 3 - 2 = 1$.

$$W_c(t_x) = \begin{cases} K - 1 & t_x \leq (M - K) \\ K - t_x - 1 & t_x > (M - K) \end{cases} \tag{27}$$

$$H_c(t_y) = \begin{cases} L - 1 & t_y \leq (N - L) \\ L - t_y - 1 & t_y > (N - L) \end{cases} \tag{28}$$

$$A(t_x, t_y) = B + C - D - E \tag{29}$$

4.1.2 Výpočet ceny překrytí bloků

Pro všechny pozice splňující podmínku, která je dána vztahem 26, je proveden výpočet ceny, která určuje podobnost bloků již umístěných ve výsledném obraze s aktuálně zpracovávaným posunem t obrazu I. Cena je dána součtem rozdílů na druhou (SSD), která je následně normalizována velikostí oblasti překrytí, mezi aktuálně zpracovávaným umístěním bloku a již umístěnými bloky do výsledného obrazu. Stejně jako u obrazu T může u obrazů S a O dojít ve vztahu pro výpočet ceny k situaci, kdy je potřeba znát hodnoty obrazu mimo rozsah oblasti $\Omega = \{(x, y)|x = 0, 1, \ldots, M - 1; y = 0, 1, \ldots, N - 1\}$. Tento problém je vyřešen nadefinováním obou obrazů mimo oblast Ω, tak že jsou rovny 0. Vztah 30 určuje cenu porovnání bloků v obraze O s obrazem I na pozici určené posunem t.

$$C(t_x, t_y) = \frac{1}{A} \sum_{x=0}^{W_c} \sum_{y=0}^{H_c} (I(x, y) - O(x + t_x, y + t_y))^2 \cdot S(x + t_x, y + t_y) \tag{30}$$

Jakmile je proveden výpočet ceny pro všechny vhodné pozice, je ze všech těchto pozic vybrána nová pozice náhodně podle pravděpodobnostní funkce (vztah 32), kde σ^2 je rozptyl hodnot pixelů vstupního obrazu I a k je parametr kontrolující náhodnost výběru pozic. Rozptyl je počítán podle vztahu 31, ve kterém n značí počet pixelů, tedy $n = K \cdot L$ a μ je průměr hodnot pixelů obrazu I. Funkce udává pravděpodobnost vybrání pozice bloku dané posunem t.

$$\sigma^2 = \frac{\sum_{x=0}^{K} \sum_{y=0}^{L} I(x, y) - \mu}{n - 1} \tag{31}$$

$$P(t_x, t_y) \propto \exp\left(-\frac{C(t_x, t_y)}{k \cdot \sigma^2}\right) \tag{32}$$

4.1.3 Vytvoření grafu pro výpočet minimálního řezu

Ve chvíli, kdy je vybrána pozice nového bloku, je potřeba zajistit nejlepší napojení nového bloku s již nakopírovanými bloky. Toto napojení je určeno pomocí řezu. K určení řezu je využita reprezentace grafem, která je následně použita pro výpočet minimálního řezu/maximálního toku. V této kapitole je popsán způsob vytvoření grafu G pro nový blok, který bude umístěn do výsledného obrazu.

Každému pixelu, který je nakopírován do výsledné textury O je přiřazeno označení i. Označení určuje, z kterého bloku pixel pochází. Jinak řečeno jakému posunu t vstupní textury I pixel odpovídá. Označení i vrcholu (pixelu) p potom je i_p. Pomocí značení vrcholů se dá výpočet minimálního řezu formulovat jako nalezení optimálního značení pixelů v oblasti překrytí.

Nejdříve jsou pro všechny pixely, které splňují podmínku $S(x + t_x, y + t_y) = 1$ vytvořeny vrcholy. Množina těchto vrcholů je značena P. Tato podmínka omezuje vytvoření grafu pouze pro pixely, u nichž dochází k překrytí mezi novým blokem a starými bloky. Počet vrcholů P je roven hodnotě $A(t_x, t_y)$, která je dána vztahem 29. Dále jsou přidány dva vrcholy nazývané terminály. Tyto speciální vrcholy jsou značeny jako zdroj a stok. Zdroj z reprezentuje již nakopírované bloky a stok s reprezentuje nový blok. Vzhledem k tomu, že v dané oblasti je možný výskyt pixelů z více než jednoho bloku, je konkrétní blok, jehož pixely mají značení i značen Z_i a nový blok je značen S_a. Jako poslední jsou do množiny všech vrcholů V přidány vrcholy pro všechny dříve vytvořené řezy mezi pixely, které se nacházejí v oblasti překrytí. Řezy se nacházejí mezi sousedícími vrcholy, kde $i_p \neq i_q$. Pokud R je množina těchto vrcholů tak je možné množinu všech vrcholů zapsat jako vztah 33.

$$V = \{P, z, s, R\} \tag{33}$$

Hodnota hrany mezi vrcholy je vypočtena pomocí vztahu 34, který udává cenu srovnání pixelů na pozici p a q dvou překrývajících se bloků A a B. Kde $\|\cdot\|$ značí L_2 normu, která je dána vztahem 9.

$$M_{cost}(p, q, A, B) = \|A(p_x, p_y) - B(p_x, p_y)\| + \|A(q_x, q_y) - B(q_x, q_y)\| \tag{34}$$

Množina dvojic sousedících pixelů je N. Dále jsou vytvořeny hrany mezi sousedícími pixely. Pro všechny sousedící pixely $(p, q) \in N$, kde $i_p = i_q$ je vytvořena hrana $e(p, q)$ nazývána n-spojení. Pokud se jedná o hranu mezi pixelem a terminálem s nebo z, je hrana nazývána t-spojení.

Dále je pro všechny sousedící pixely $(p, q) \in N$, kde $i_p \neq i_q$, vytvořena trojice hran $e(p, r)$, $e(r, q)$ a $e_s(r)$. Vrchol $r \in R$ je vrcholem určujícím již vytvořený řez v obraze. Hrany $e(p, r)$ a $e(r, q)$ vytvářejí n-spojení mezi vrcholem řezu a pixely mezi nimiž je řez umístěn. Hrana $e_s(r)$ je t-spojení mezi terminálem s a vrcholem řezu.

Jako poslední je potřeba vytvořit t-spojení mezi terminály a některými pixely. Hraně $e_z(p)$ nebo $e_s(p)$, která bude t-spojením dvou vrcholů, kde jeden z vrcholů bude terminál z nebo s a druhý bude pixelem $p \in P$, je přiřazena nekonečně vysoká hodnota. Přiřazením nekonečné hodnoty, je pro dané pixely určeno, z kterého bloku budou nakopírovány. V případě t-spojení s terminálem s budou pixely pocházet z nového bloku a v případě t-spojení s terminálem z zůstanou ve výsledném obraze již nakopírované pixely.

Nyní je potřeba specifikovat pixely, pro které budou vytvořeny hrany $e_z(p)$ a $e_s(p)$. Všechny pixely $p \in P$ pro které platí, že pro alespoň jeden sousední pixel nacházející se uvnitř oblasti překrytí je hodnota v obraze S rovna 0, jsou spojeny s terminálem s hranou $e_s(p)$. Pokud má pixel méně než čtyři sousední pixely nacházející se uvnitř oblasti překrytí a zároveň je hodnota všech těchto sousedních pixelů v obraze S rovna 1, potom je tento pixel spojen s terminálem z hranou $e_z(p)$. Tabulka 1 ukazuje přiřazení hodnot pro všechny zmíněné hrany.

Hrana	Hodnota	Pro
$e_s(p)$	∞	$p \in P$
$e_z(p)$	∞	$p \in P$
$e(p, r)$	$M_{cost}(p, q, S_a, Z_{i_q})$	$r \in R \wedge p \in P$
$e(r, q)$	$M_{cost}(p, q, Z_{i_p}, S_a)$	$r \in R \wedge q \in P$
$e_s(r)$	$M_{cost}(p, q, Z_{i_p}, Z_{i_q})$	$r \in R$
$e(p, q)$	$M_{cost}(p, q, Z_{i_p}, S_a)$	$p \in P \wedge q \in P$

Tabulka 1: Určení hodnot pro jednotlivé hrany vznikající při vytváření grafu.

Popsaný způsob vytvoření grafu a formulace nalezení řezu jsou shodné s popisem problematiky v článku [23] a odpovídá α-rozšíření. V článku také zmiňují, že výpočet hodnot pro jednotlivé hrany musí být metrikou. Z tohoto důvodu byla pro výpočet zvolena L_2 norma. Tato podmínka je zavedena, aby při přerušení jedné hrany spojené s vrcholem řezu bylo toto přerušení levnější než přerušení zbývajících dvou hran najednou. Toto je zaručeno metrikou splňující trojúhelníkovou nerovnost. Pro důkaz, že nalezení minimálního řezu v takto vytvořeném grafu, je optimálním napojením mezi bloky, je čtenář odkázán na již zmiňovaný článek.

4.1.4 Nalezení minimálního řezu

Algoritmus k nalezení minimálního řezu grafem naimplementovaný pro tuto práci odpovídá algoritmu uvedenému v článku [24]. Autoři článku uvádějí, že tento algoritmus je rychlejší než většina běžně používaných algoritmů řešících tuto problematiku. Nalezení řezu nad grafem popsaným v předchozí kapitole spadá do kategorie problematiky zjišťování minimálního řezu a maximálního toku v síti. Použitý algoritmus patří mezi metody zvětšování cest. Doplňující informace k sítím jsou v textu [25].

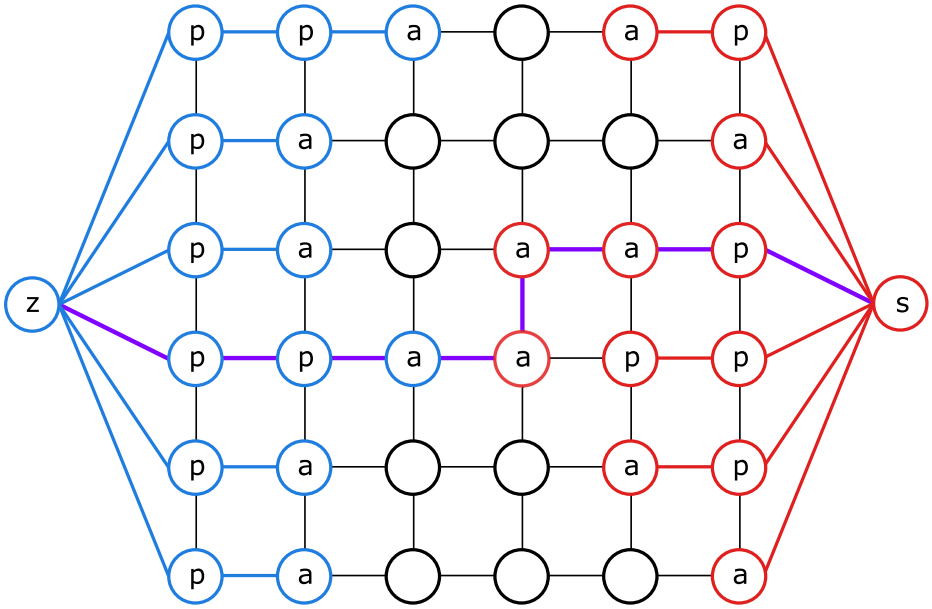
Pro nalezení řezu je potřeba určit, které vrcholy v grafu $G = \langle V, E \rangle$ jsou zdroje a stoky v síti. Zdrojem v síti je vrchol z a stokem vrchol s. Množina E jsou všechny ohodnocené hrany vytvořené podle pravidel v předchozí kapitole. Na základě grafu G je vytvořen residuální graf G_r, jehož hrany jsou ohodnoceny rezervami kapacit odpovídajících hran v grafu G. Struktura grafu G je shodná se strukturou grafu G_r.

Dále jsou pro výpočet uchovávány dva stromy Z a S sloužící k vyhledávání cest ze zdroje do stoku. Kořenem stromu Z je zdroj z a kořenem stromu S je stok s. Tyto stromy se nepřekrývají, aby nedošlo k situaci, kdy jeden vrchol náleží oběma stromům. Při nalezení dvou vrcholů p, q spojených hranou $e(p, q)$, kde p je součástí jiného stromu než q, dochází k nalezení cesty umožňující zvětšení toku v síti.

Jednotlivé vrcholy grafu můžou nabývat tří různých stavů měnících se v průběhu výpočtu. Vrchol může být pasivní, aktivní nebo volný. Pasivní stav mají vrcholy, které neumožňují další zvětšení stromu. Jsou to vrcholy, jejichž všichni sousedi jsou ve stavu aktivní nebo pasivní, neboli všichni sousedi jsou součástí stejného stromu. Pokud je vrchol aktivní, znamená to, že minimálně jeden z jeho sousedů je ve stavu volný. Aktivní vrcholy slouží k připojení sousedních volných vrcholů k jejich stromu. Posledním typem vrcholů jsou volné vrcholy. Tyto vrcholy nepatří k žádnému stromu a umožňují zvětšování stromů. Příklad ukazující jednotlivé stavy vrcholů je na obrázku 19.

Algoritmus provádí tři základní kroky, které opakuje, dokud v grafu existuje nenasycená cesta P ze zdroje z do stoku s. V případě, že taková cesta neexistuje, je nalezen maximální tok v síti a stromy Z, S jsou rozděleny nasycenými hranami $e(p, q)$ tvořícími minimální řez, kde $p \in Z$, $q \in S$ nebo naopak. K vypočtení řezu je potřeba dvou pomocných množin. První je množina A uchovávající seznam aktuálně aktivních vrcholů. Druhou je množina O uchovávající aktuální seznam vrcholů označených jako sirotci. Jak se z vrcholu stane sirotek je uvedeno dále v textu při popisu poslední fáze algoritmu. Jednotlivé fáze jsou zvětšení stromu Z nebo S pro nalezení cesty P, zvětšení toků podél nalezené cesty a adoptování sirotků v množině O. Jednotlivé množiny vypadají na začátku algoritmu takto.

$$Z = \{z\}, \quad S = \{s\}, \quad A = \{z, s\}, \quad O = \emptyset \tag{35}$$



Obrázek 19: Ukázka stromů Z a S a stavů vrcholů. Červeně jsou ohraničeny vrcholy patřící do stromu S a modře jsou ohraničeny vrcholy patřící do Z. Nalezená cesta P ze zdroje z do stoku s je vyznačena fialovou barvou. Aktivní vrcholy jsou značeny písmenem a. Pasivní vrcholy mají písmeno p. Černé ohraničení vrcholů je použito pro volné vrcholy.

Při zvětšení stromů Z nebo S dochází k připojování volných vrcholů k jednotlivým stromům, dokud není nalezena nenasycená cesta P ze zdroje z do stoku s. Postupně jsou procházeny vrcholy z množiny A. Pro každý vrchol jsou prozkoumáni všichni jeho sousedi spojení hranou, jejíž rezerva kapacity je větší než 0. Jedná-li se o volný vrchol, je připojen k danému stromu a stává se z něj aktivní vrchol. Poté, co jsou všichni tito sousedi prozkoumáni, je aktuálně zpracovávaný vrchol odebrán z množiny A a stává se pasivním. V případě nalezení cesty P, je tato fáze algoritmu přerušena a cesta je dále zpracována v další fázi. Cesta je nalezena, pokud mezi zkoumaným vrcholem a jeho sousedem existuje nenasycená cesta a oba vrcholy patří do jiného stromu.

$$Strom(p) = \begin{cases} Z & p \in Z \\ S & p \in S \\ \emptyset & p \notin Z \wedge p \notin S \end{cases} \tag{36}$$

Následující pseudokód popisuje fázi zvětšování stromů. V popisu je použito značení $Strom(p)$, které určuje do kterého stromu vrchol p patří. Značení je dáno vztahem 36.

```
while A ≠ ∅
    vezmi vrchol p ∈ A
    for všechny sousedy q, kde rezerva kapacity hrany e(p,q) > 0
        if q ∉ Z ∧ q ∉ S
            p je nyní rodičem q, A = A ∪ {q}
            if p ∈ Z
                Z = Z ∪ {q}
            if p ∈ S
                S = S ∪ {q}
        if (q ∈ Z ∨ q ∈ S) ∧ Strom(p) ≠ Strom(q)
            return nenasycenou cestu P ze z do s
end while
return P = ∅
```

V následující fázi probíhá zpracování nalezené cesty P. Nejdříve je zjištěna hodnota c, která je rovna nejmenší rezervě kapacit hran podél cesty. Tato hodnota udává, o kolik je možné celkový tok v síti navýšit. Poté je pro každou hranu (p, q) v grafu G_r podél cesty P odečtena od rezervy kapacity zjištěná hodnota c. Při tomto aktualizování grafu může dojít k nasycení některých hran. Pokud je nasycená hrana mezi vrcholy, které patří do stejného stromu, stává se jeden z vrcholů sirotkem a je přidán do množiny O. Vrchol se stává sirotkem, protože v daném případě je zrušeno jeho spojení s rodičem. Postup pro fázi zpracování cesty P je následující.

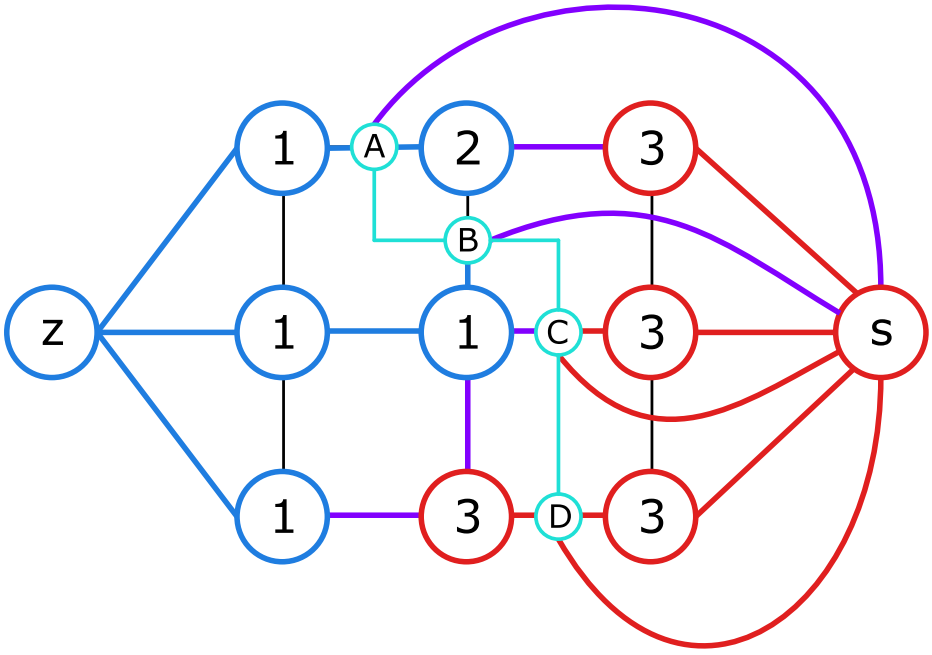
```
najdi nejmenší rezervu kapacit hran c podél cesty P
zmenši rezervu kapacit hran podél cesty P v grafu G_r o c
for všechny nasycené hrany e(p,q) ∈ P
    if Strom(p) = Strom(q) = Z
        zruš rodiče vrcholu q, O = O ∪ {q}
    if Strom(p) = Strom(q) = S
        zruš rodiče vrcholu p, O = O ∪ {p}
end for
```

Poslední fází algoritmu je zpracování všech vrcholů, které se staly sirotky. Pro vrchol $o \in O$ je v této fázi hledán nový rodič. Pokud je rodič nalezen zůstává vrchol součástí aktuálního stromu. Avšak pokud není nalezen vhodný rodič vrcholu o je jeho stav nastaven na volný. Aby se vrchol p mohl stát novým rodičem vrcholu o musí splňovat několik podmínek. První podmínkou je, že vrcholy p a o musí být ve stejném stromě ($Strom(p) = Strom(o)$). Dále musí být hrana (p, o) nenasycená a nakonec musí být původem vrcholu p zdroj nebo stok. Původem je zde myšleno, že žádný vrchol na cestě z p do z nebo z p do s nesmí být sirotkem. V případě nalezení vrcholu splňujícího podmínky se p stává novým rodičem a vrchol o zůstává ve stejném stromě. Při nenalezení rodiče splňujícího dané podmínky, je zrušena příslušnost vrcholu o k danému stromu a všichni jeho potomci jsou přidáni do množiny O (stávají se sirotky).

Vzhledem k tomu, že se vrchol o stává volným je potřeba změnit stav pro všechny jeho sousedy spojené nenasycenou hranou na aktivní.

Algoritmus končí ve chvíli, kdy neexistuje žádná nenasycená cesta P ze zdroje z do stoku s. Jakmile je algoritmus ukončen je možné určit, které pixely nového bloku mají nahradit již nakopírované pixely. Všechny vrcholy stromu S odpovídají pixelům, které mají být nahrazeny odpovídajícími pixely z nového bloku. Dále je možné u vrcholů řezů $r \in R$ určit, co se s řezy stane při určování nového značení pro nový blok. Pokud je přerušena hrana $e_s(r)$ znamená to, že původní řez zůstává. Úplné zrušení řezu v místě vrcholu r nastává v případě, že ani jedna z trojice hran vedoucích k danému vrcholu řezu není součástí minimálního řezu. A v případě přerušení jedné z hran $e(p, r)$, $e(r, q)$ je řez nahrazen řezem o menší ceně.

Na obrázku 20 je ukázán výsledek nalezení minimálního řezu. Jednotlivé vrcholy jsou značeny číslem určujícím bloky, z kterých byl pixel zkopírován. V příkladu na obrázku je označení nově kopírovaného bloku číslo $i = 3$. Jednotlivé vrcholy původního řezu ukazují různé případy, které mohou pro tyto řezy nastat. Například vrchol řezu A má přerušenou hranu se stokem. To znamená, že tento řez zůstává ve výsledném obraze. Vrchol B je stejným případem jako vrchol A. Přesným opakem je vrchol D, který nemá přerušenou žádnou hranu a proto je z výsledného obrazu vymazán. Posledním případem je přerušení jedné ze dvou hran spojujících vrchol řezu s pixely (vrchol C). Při přerušení této hrany je původní řez na dané pozici nahrazen novým řezem a cena tohoto řezu je dána cenou přerušené hrany.

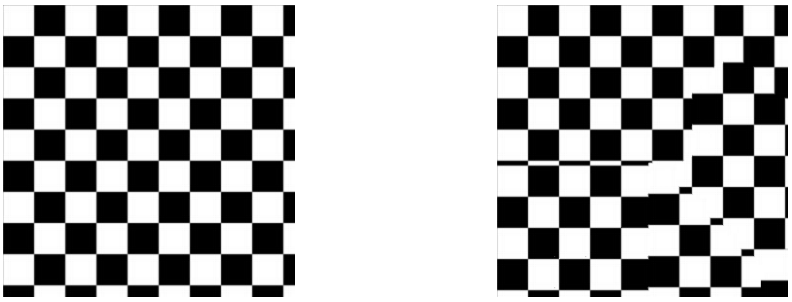


Obrázek 20: Ukázka nalezení minimálního řezu v grafu G. Vrcholy stromu Z jsou značeny modře a vrcholy stromu S červeně. Fialově jsou označeny hrany, které tvoří nový řez. Tyrkysové značení je použito pro řez nacházející se v oblasti. Tyrkysové hrany mezi vrcholy nepatří do grafu G. Čísla vrcholů reprezentují číslo bloku, z kterého pochází hodnota daného pixelu.

4.1.5 Parametr k

Parametr k ze vztahu 32 ovlivňuje výběr vhodného posunu t udávajícího pozici vstupního obrazu I ve výstupním obraze O. Hodnota ovlivňuje pravděpodobnost výběru dané pozice. Při zvětšení hodnoty k dochází k zvětšení pravděpodobnosti výběru pozice a naopak při zmenšení dochází k zmenšení pravděpodobnosti výběru pozice. Kromě ovlivnění samotné pravděpodobnosti výběru pozice, má parametr také vliv na náhodnost výběru vhodné pozice, která nejlépe zapadá do výsledného obrazu. Malé hodnoty parametru k mají za následek striktní výběr vhodných pozic. Jsou vybírány přednostně ty pozice, které mají nízkou cenu C. Při vyšších hodnotách parametru k je zvýšena pravděpodobnost výběru pozic, které mají vyšší cenu C. Na následujícím příkladu je demonstrován vliv tohoto parametru.

Pro zjednodušení jsou porovnány výsledky pouze pro dvě pozice. Cena první pozice je $C_1 = 10$ a cena druhé pozice je $C_2 = 50$. Rozptyl hodnot ukázkového obrazu je $\sigma^2 = 5000$. Nejdříve jsou vypočteny pravděpodobnosti těchto pozic pro parametr $k = 1$. Podle vztahu 32 je pravděpodobnost první pozice $P_1 = 0.998$ a pravděpodobnost druhé pozice je $P_2 = 0.991$. Z výsledků je vidět, že i když je cena druhé pozice pětkrát větší než cena první pozice, není rozdíl v pravděpodobnostech výrazný. Pro demonstraci malého rozdílu v těchto pravděpodobnostech je proveden výpočet ještě pro parametr $k = 0.01$. Pro tento parametr vyjde pravděpodobnost první pozice $P_1 = 0.819$ a pravděpodobnost druhé pozice $P_2 = 0.367$. Jak je zřejmé z těchto výsledků při menší hodnotě parametru k se zvětšuje rozdíl mezi pravděpodobnostmi pozic s nízkými cenami a pravděpodobnostmi pozic s vysokými cenami. To má za následek ovlivnění náhodnosti výběru pozic podle jejich ceny.



(a) $k = 10$ (b) k=0.001

Obrázek 21: Ukázka vlivu parametru k na výsledek syntézy.

4.1.6 Vylepšování

Výběr vhodné pozice pro umístění nového bloku do výsledného obrazu O popsaný v kapitole 4.1.1 platí pouze pro inicializaci obrazu O. Inicializace probíhá, dokud pro obraz T nadefinovaný v zmíněné kapitole není splněna podmínka daná vztahem 37, kde M a N jsou rozměry obrazu O a $m = (M-1)$, $n = (N-1)$. Touto podmínkou je zaručeno zaplnění celého výsledného obrazu O, protože obraz T je integrálním obrazem masky S. Zaplněním celého obrazu je myšleno nakopírování alespoň jednoho pixelu ze vzorové textury do všech pixelů výsledného obrazu.

$$T(m,n) = M \cdot N \tag{37}$$

Jakmile je provedena inicializace výsledného obrazu, je možné dále vylepšovat syntetizovaný výsledek. Syntetizovaný výsledek je vylepšován na základě provedených řezů při navazování bloků. Aby bylo takové vylepšování možné, je potřeba výsledky všech provedených řezů ukládat. Ukládání těchto hodnot řezů je realizováno pomocí obrazu R_e o rozměrech $P \times Q$, kde $P = 2M-1$ a $Q = 2N-1$. Indexování řezu v $R_e(x,y)$ mezi dvěma pixely na pozicích (x_1, y_1), (x_2, y_2) v O je provedeno podle vztahu 38.

$$\begin{aligned} x &= \frac{x_1 + x_2}{2} \\ y &= \frac{y_1 + y_2}{2} \end{aligned} \tag{38}$$

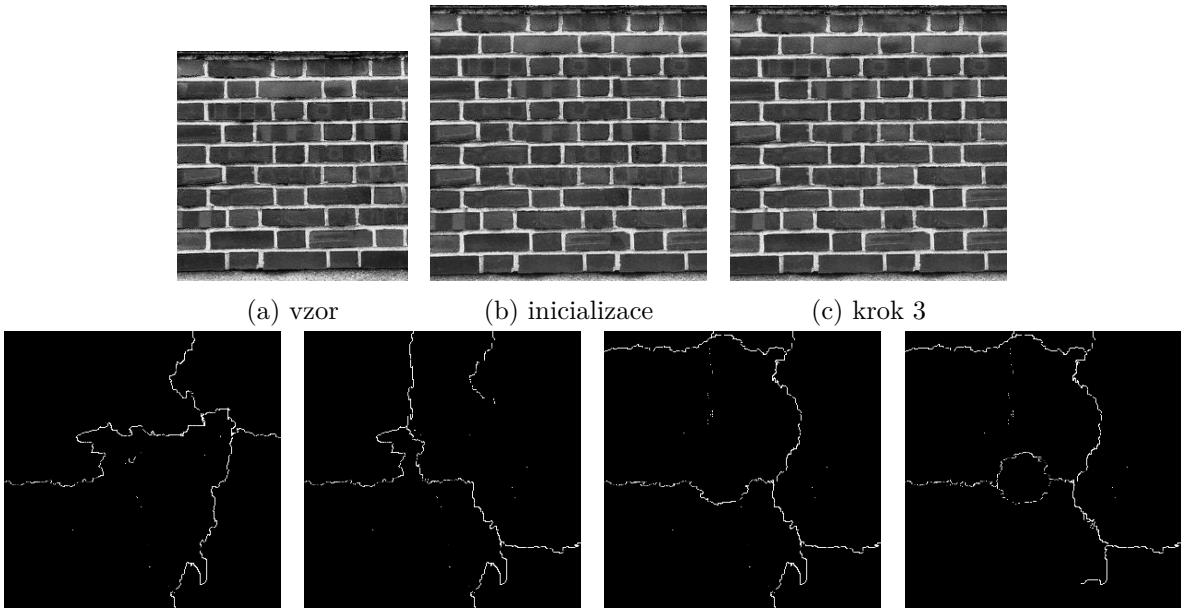
Při vylepšování je vybírána oblast v obraze O, u které je nejpravděpodobnější výskyt chyb. Tyto chyby vytvářejí viditelné řezy v syntetizované textuře a vznikají špatným navázáním mezi bloky. Pro zjištění oblasti, která potřebuje nejvíce vylepšit, je vypočítán součet hodnot všech řezů nacházejících se v dané oblasti. K zrychlení výpočtu součtů pro různé oblasti je použit integrální obraz. Integrální obraz obrazu R_e je značen R_i a má stejné rozměry. Hodnoty obrazu R_i mimo jeho oblast jsou rovny 0. Hodnota E_r určuje chybovost řezů v dané oblasti o rozměrech $R \times S$. Rozměry R a S jsou v této práci zvoleny následovně $R = M/2$, $S = N/2$. Chybovost řezů je počítána pro pozice (x,y) v obraze O, kde rozsah hodnoty x je $0, \ldots, (M-R)$ a hodnoty y je $0, \ldots, (N-S)$. Vztahy 39 udávají výpočet chybovosti E_r pro oblast na pozici (x,y).

$$\begin{aligned} A &= R_i(2x-1, 2y-1) \\ B &= R_i(2x+2R-1, 2y-1) \\ C &= R_i(2x-1, 2y+2S-1) \\ D &= R_i(2x+2R-1, 2y+2S-1) \\ E_r(x,y) &= A + D - C - B \end{aligned} \tag{39}$$

Po vypočtení hodnoty E_r pro všechny oblasti je vybrána ta oblast, která má danou hodnotu největší. Následně je vypočtena cena a pravděpodobnost pro všechny pozice vstupní textury I, které překrývají celou oblast pro výpočet chybovosti. Cena a pravděpodobnost jsou vypočítány podle vzorců 30 a 32. Pozice(x, y) jsou vybírány v rozsahu $x = x_e - R, \ldots, x_e$ a $y = y_e - R, \ldots, y_e$. Hodnoty x_e,y_e udávají pozici oblasti vybrané pro vylepšení.

Ze všech pozic vstupního obrazu I splňujících dané podmínky je na základě jejich pravděpodobnosti náhodně vybrána jedna. Pro vybranou pozici je vytvořena reprezentace grafem popsaná v kapitole 4.1.3 a poté je proveden výpočet minimálního řezu. Popis výpočtu minimálního řezu je v kapitole 4.1.4. Poté je na základě minimálního řezu nakopírován nový blok do výsledné textury O a jsou aktualizovány hodnoty hran v obraze R.

Krok popsaný v této kapitole je možné opakovat a počet opakování není omezen.



(a) vzor (b) inicializace (c) krok 3

(d) inicializace (e) krok 1 (f) krok 2 (g) krok 3

Obrázek 22: Ukázka jednotlivých kroků při vylepšování řezů v inicializované textuře a výsledku po třech krocích vylepšování.

4.2 Zrychlení

Zjišťování ceny překrytí pomocí SSD popsané v kapitole 4.1.2 je časově náročné. Například vygenerování textury o rozměrech 384x384 pixelů ze vzorové textury o velikosti 192x192 pixelů trvá přibližně 12 minut. Podle článku [4] je výpočet syntézy možné urychlit využitím odlišného vzorce pro výpočet ceny překrytí.

Namísto použití vztahu 30 pro výpočet ceny překrytí lze použít vztah 40.

$$\begin{aligned} C(t_x, t_y) = \sum_{x=0}^{W_c} \sum_{y=0}^{H_c} I(x,y)^2 + \sum_{x=0}^{W_c} \sum_{y=0}^{H_c} O(x+t_x, y+t_y)^2 - \\ - 2 \sum_{x=0}^{W_c} \sum_{y=0}^{H_c} I(x,y) \cdot O(x+t_x, y+t_y) \end{aligned} \tag{40}$$

První a druhou sumu je možné vypočítat pomocí integrálního obrazu a poslední suma je konvolucí obrazu I a O. Použitím vztahu 29 sloužícího pro výpočet sumy v integrálním obraze a vztahu 18 pro výpočet konvoluce je vztah 40 přepsán na vztah 41. Při výpočtu A_O je použit vztah 29 uvedený v kapitole 4.1.1 s jednou výjimkou a to, že integrální obraz T ve vztahu je nahrazen integrálním obrazem hodnot obrazu O umocněných na druhou. Stejně jako pro výpočet A_O je pro A_I použit integrální obraz hodnot obrazu I umocněných na druhou. V případě výpočtu A_I jsou hodnoty W_c a H_c dány vztahy 24, 25. Pokud je hodnota $t_x < 0$ potom $x_i = -t_x$ jinak $x_i = 0$. Stejně je určena hodnota y_i.

$$C(t_x, t_y) = A_I(x_i, y_i) + A_O(x+t_x, y+t_y) - 2(I * O)(t_x, t_y) \tag{41}$$

Výpočet konvoluce je dán vztahem 18, kde rozměry masky W, H jsou dány vztahy 24, 25 a rozsah hodnot m,n je $0, \ldots, W$; $0, \ldots, H$. Při výpočtu konvoluce může dojít k vyžadování hodnot mimo oblast obrazů O a I, proto jsou oba obrazy vně nadefinované oblasti rovny 0. Konvoluce mezi obrazy je počítána pomocí rychlé Fourierovy transformace (FFT). Využitím integrálních obrazů a FFT se podařilo zrychlit generování textur z 12 minut při použití SSD výpočtu na přibližně 50 vteřin.

V případě vztahu 41 bylo zjištěno, že nahrazením členu $A_I(x_i, y_i)$ za $A_I(K, L)$, kde $K \times L$ jsou rozměry obrazu I, je dosaženo lepších výsledků. Při generování výsledků bylo také zjištěno, že metoda využívající upravený vzorec generuje dobře pouze stochastické textury. V případě ostatních kategorií textur daný vzorec neumožňuje správné umístění nových bloků. Pro použití takto upraveného vzorce je potřeba předem znát ohraničení oblasti překrytí bloků a to v případě použitého výběru umístění nových bloků není možné. Proto nelze urychlení použít při inicializaci textury, avšak lze jej použít pro fázi vylepšování, kdy je předem dané ohraničení překrývajících se oblastí.

5 Výsledky

Pro generování textur byla použita vlastní implementace metod Image Quilting a Graphcuts. Implementace metod odpovídá popisu uvedenému v kapitolách 3 a 4. Dále bylo kromě dvou popsaných metod, provedeno také srovnání s upravenou verzí metody Image Quilting. Popis úpravy využívající detekci hran je možné nalézt v kapitole 3.3. Porovnání výsledků jednotlivých metod bylo provedeno na základě vygenerovaných textur. Odkazy na datasety obsahující textury použité pro porovnávání je možné nalézt v příloze na CD. Ukázky některých vygenerovaných textur jsou v příloze.

K vytvoření výsledné aplikace umožňující generování textur pomocí dříve zmíněných metod bylo použito vývojové prostředí Microsoft Visual Studio 2013. Samotná implementace byla napsána v jazyce C++ s využitím knihovny OpenCV vydané pod BSD licencí. A uživatelské rozhraní aplikace bylo vytvořeno pomocí frameworku Qt, který je k dostání pod licencí GPL. Použitá verze OpenCV je 3.0.0 a verze frameworku Qt je 5.5.1.

Z pohledu uživatele je mezi metodami značný rozdíl a to v možnostech volby parametrů ovlivňujících samotný proces syntézy. V případě metody Graphcuts je volen pouze jeden další parametr (parametr k) spolu s velikostí výsledné textury, která je určována pro všechny tři testované metody. Avšak metoda Image Quilting umožňuje větší kontrolu nad samotným výsledkem syntézy pomocí tří volitelných parametrů (velikost bloku, velikost překrytí, parametr k). Při využití detekce hran jsou tyto parametry rozšířeny ještě o možnosti detekce hran a filtrace obrazu, které jsou popsány v kapitole 3.3. Pro efektivní využití větší kontroly nad metodou Image Quilting a její upravenou verzí je potřeba, aby uživatel alespoň základně rozuměl vlivu jednotlivých parametrů na výsledek syntézy (kapitola 3.2).

Dále je potřeba zmínit, že u všech zmíněných metod jsou při syntéze některé části algoritmu náhodné. A proto při opakovaném spuštění jedné metody se stejně nastavenými parametry pro stejnou vzorovou texturu není výsledek syntézy vždy stejný.

5.1 Srovnání metod Image Quilting a Graphcuts

Při porovnávání je předpokládáno správné nastavení parametrů pro vygenerování textury. Výsledek syntézy je porovnáván z pohledu shody vzorů ve vzorové textuře se vzory v syntetizované textuře. Dále je brán v potaz výskyt tzv. artefaktů. Artefakty jsou chyby vzniklé při generování textury, které se projevují viditelným přerušením vzorů při navazování bloků. Pro rozdělení textur do kategorií je použito poslední dělení uvedené v kapitole 1.3.

V případě regulárních (regular) textur jsou výsledky generované oběmi metodami dobré. Metody dodržují vzory ve vzorové textuře a výskyt artefaktů je velice ojedinělý. Většinou je způsoben náhodností, která je součástí obou metod. Pokud se u regulární textury vyskytne nějaký artefakt, většinou stačí spustit znovu syntézu se stejným nastavením.

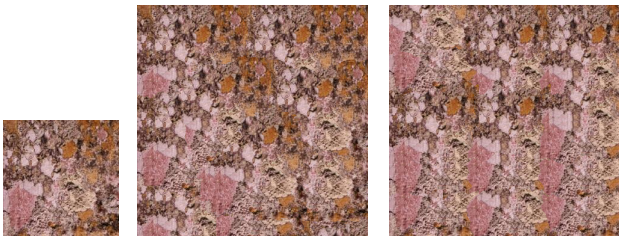
Výsledky pro strukturované (structured) textury jsou podobné výsledkům u regulárních textur s tím rozdílem, že u strukturovaných se častěji vyskytují artefakty a to hlavně v případě metody Image Quilting. Častější výskyt artefaktů je způsoben nepravidelnou velikostí prvků v textuře.

Pro textury z kategorie buňkových (cellular) je možné říct, že výsledky generování jednotlivými metodami jsou srovnatelné a to jak v dodržování vzorů tak ve výskytu artefaktů.

Přesto, že jsou výsledky těchto metod pro předchozí tři kategorie srovnatelné, lze u kategorie částečně strukturovaných/stochastických (semi-structured/stochastic) textur říct, že výsledky metody Graphcuts jsou lepší než výsledky metody Image Quilting. To je způsobeno neumisťováním bloků do předem připravených pozic a přesnějším výpočtem řezu mezi bloky. Díky náhodnému uspořádání prvků v textuře je v této kategorii výskyt artefaktů méně znatelný než například v kategorii regulárních textur.

Poslední kategorií jsou textury obsahující velké prvky (large features). V této kategorii značně převyšuje metoda Graphcuts metodu Image Quilting. Ukázka výsledků pro tuto kategorii je na obrázku 23. Stejně jako v kategorii strukturovaných/stochastických textur je tento fakt způsoben komplikovanějším výpočtem řezu a jiným přístupem k umisťování bloků do syntetizované textury. Kvůli komplikovanosti vzorů textur v této kategorii nelze určit četnost výskytu artefaktů z důvodu jejich špatného rozpoznání v textuře.

U všech zmíněných kategorií existují konkrétní textury, pro které jedna z metod generuje lepší výsledky. A proto nelze říct, že zmíněné výsledky metod pro jednotlivé kategorie platí pro všechny textury z dané kategorie, ale platí pouze obecně. Jeden takový příklad je na obrázku 25.



(a) vzor (b) Graphcuts (c) Image Quilting

Obrázek 23: Ukázka výsledku pro textury s velkými prvky vygenerovaných metodou Image Quilting (c) a metodou Graphcuts (b) ze vzorové textury (a).

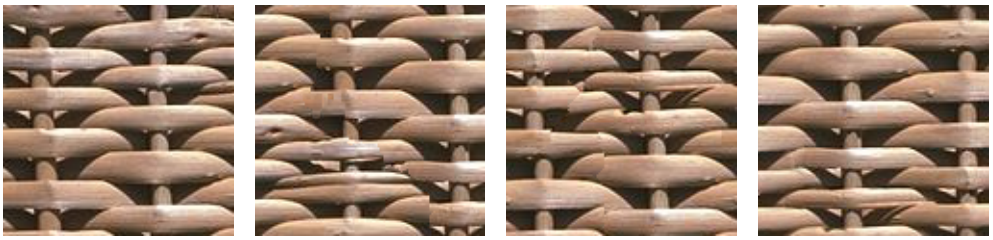
(a) vzor (b) Graphcuts (c) Image Quilting

Obrázek 24: Ukázka výsledku strukturovaných textur demonstrující konkrétní příklad, kdy pro kategorii, kde jsou většinou výsledky stejné, je znatelný rozdíl mezi výsledky dvou metod.

5.2 Využití detekce hran pro Image Quilting

Image Quilting s využitím detekce hran v porovnání s neupravenou metodou Image Quilting generuje ve většině případů horší výsledky. V porovnání s metodou Graphcuts je v kategoriích částečně strukturovaných/stochastických textur a textur s velkými prvky tato upravená metoda horší. Avšak ve zbývajících dvou kategoriích regulárních a strukturovaných textur existují speciální případy, kdy dosahuje Image Quilting s využitím detekce hran lepších výsledků než druhé dvě metody.

Těmito speciálními případy jsou zejména textury, kde je velmi malá variace v barvě jednotlivých prvků. Příkladem vzoru s takovými prvky je obrázek 25a. V takovém případě je zachování správného napojení jednotlivých prvků důležitější než barevná informace. Proto je použita detekce hran a syntéza je provedena na základě této detekce hran. Důvodem selhání druhých dvou metod je barevná podobnost jednotlivých prvků, které jsou následně špatně navázány a vytvářejí chyby v textuře. K špatnému navázání dochází při výpočtu podobnosti bloků, kde rozmezí hodnot určujících podobnost bloků je příliš malé a následně je vybrán špatný blok.



(a) vzor (b) Graphcuts (c) Image Quilting (d) Detekce hran

Obrázek 25: Porovnání chyb při využití detekce hran. Jednotlivé obrázky ukazují chyby vzniklé při syntéze pro jednotlivé metody.

K zlepšení výsledků metody Image Quilting využívající detekci hran by bylo možné pro výpočet podobnosti bloků (vztah 10) využít zároveň informace o barvě obsažené ve vzorové textuře a informace získané detekcí hran. Následně by pro výběr vhodných nových bloků mohl být použit například součet chyby překrytí pro vzorovou textura a chyby překrytí pro obraz reprezentující hrany v této textuře.

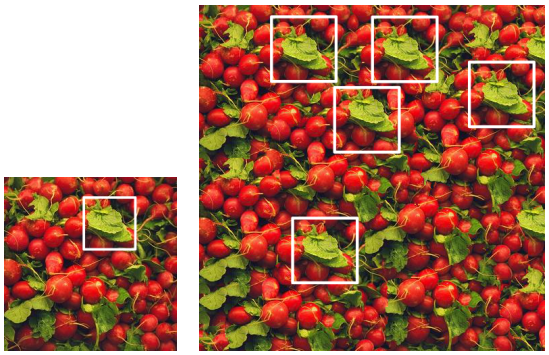
6 Problémy a jejich možná řešení

V této kapitole jsou popsány problémy, ke kterým dochází při syntéze textur pomocí metod Image Quilting a Graphcuts. Následně jsou navrhnuta možná řešení těchto problémů.

Při využívání textur se často stává, že textura, kterou je potřeba nanést na povrch objektu má příliš malé rozměry. Jednou ze základních možností, jak takovou situaci vyřešit, je použít opakování textury. Avšak tento přístup ve většině případů vede k viditelnému opakování jedné textury poskládané vedle sebe. Tento problém je možné vyřešit právě syntézou textur, kterou je vygenerována textura o dostatečném rozlišení tak, aby nebylo znatelné opakování vzorové textury. Nyní jsou popsány situace, kdy u syntetizované textury dochází k viditelnému opakování vzorové textury.

6.1 Vzorové textury

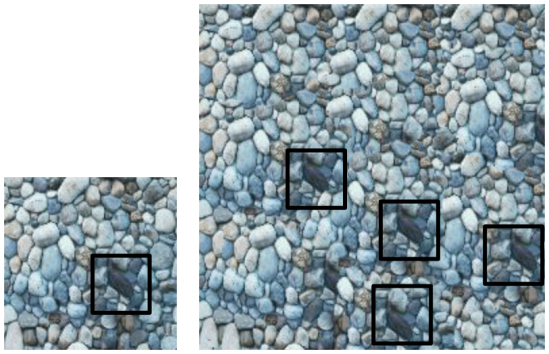
Prvním problémem, který může vést k špatně syntetizované textuře, je výskyt specificky tvarovaných objektů ve vzorové textuře. Při opakování takového objektu v syntetizované textuře je hned na první pohled jasné, že textura byla vygenerována kopírováním původní textury. Takto vzniklý výsledek je u syntetizovaných textur považován za špatný. Na obrázku 26a je vyznačena ukázka popisovaného typu objektů. A na obrázku 26b je ukázka vygenerované textury ze vzorové textury obsahující specificky tvarovaný objekt.



(a) (b)

Obrázek 26: Opakování specificky tvarovaných objektů při syntéze textury.

Další problém je podobný tomu předchozímu. Jedná se o výskyt kontrastních objektů ve vzorové textuře. V případě opakování takového vzoru v syntetizované textuře je jasně zřetelné vytvoření textury na základě jejího opakování. U některých textur se nemusí jednat pouze o kontrastní objekt, ale například o kontrastní oblasti. Avšak výskyt takových oblastí není příliš častý. Na obrázku 27a je vzorová textura obsahující kontrastní objekt a na obrázku 27b je možné vidět výsledek syntézy pro danou vzorovou texturu.



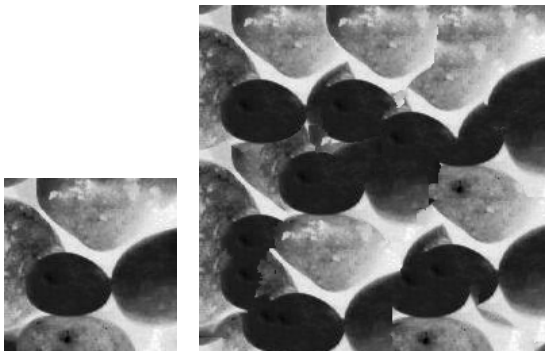
(a) (b)

Obrázek 27: Kontrastní objekt ve vzorové textuře a ukázka výsledku syntetizované textury.

Oba popsané problémy by mohly být pro metodu Image Quilting vyřešeny například označením objektu, který by při opakování způsobil špatný výsledek syntézy a vyřazením vyznačené oblasti z bloků kopírovaných do generované textury. Označení by prováděl uživatel a mohlo by vypadat například jako na obrázku 26a.

U metody Graphcuts by se dal problém řešit stejným způsobem. Označená část by byla vyřazena z počítání ceny překrytí bloků a následně i z kopírování do výsledné textury.

Jedním z hlavních problémů je nedostatek variací ve vzorové textuře. Nedostatek variací způsobuje nemožnost správného navázání bloků. Jediným možným řešením je použití textury s dostatkem variací pro syntézu textur. Například na obrázku 28a jsou oválné tvary, ale pouze jeden objekt je celý. U ostatních objektů zcela chybí jejich části potřebné pro nakopírování těchto objektů do syntetizované textury.



(a) vzor (b) Graphcuts

Obrázek 28: Vzorová textura (a) s nedostatkem variací pro syntézu (b).

6.2 Seamless textury

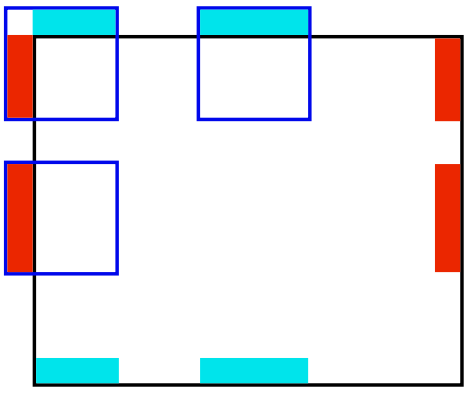
Pokud jsou malé rozměry textury řešeny jejím opakováním, dochází v místech, kde se dotýkají hrany textur k viditelným přechodům mezi opakovanými texturami. Ve většině případů se tento problém řeší použitím textur, jejichž hrany na sebe navazují (seamless texture). V případě, že hrany textury na sebe navazují a její pravidelné opakování nezpůsobí nereálný dojem, lze tuto texturu použít.

Avšak pokud na sebe hrany nenavazují, je jednou z možných metod řešení rozdělení textury na čtyři části. Tyto části jsou následně proházeny tak, aby se jejich vnitřní okraje staly vnějšími okraji textury. Poté je potřeba zahladit napojení těchto částí na hranách, které se nyní nacházejí uvnitř textury. Tento postup je komplikovaný a vyžaduje značnou zručnost osoby, která tento proces provádí.

Samotná syntéza tento zmíněný případ řeší vygenerováním dostatečně velké textury, aby nebylo potřeba použít opakování jedné textury. Ale existují případy, kdy nezáleží na velikosti textury a je potřeba, aby na sebe okraje navazovaly. Takovým příkladem může být například textura pro povrch válce.

I tento problém je možné vyřešit při syntéze textur. Pro metodu Image Quilting by bylo potřeba přidat konkrétní podmínky. První podmínkou je, aby bloky v prvním řádku a prvním sloupci přesahovaly okraj textury o velikost překrytí bloků. Části přesahující okraj je potřeba umístit na příslušné pozice do posledního sloupce a posledního řádku. Při umísťování bloků do posledního sloupce a posledního řádku by byl nový blok vybírán na základě podobnosti s již nakopírovanými bloky a s částí patřící bloku umístěnému na opačném okraji textury. Následný výpočet řezu by proběhl navíc i v oblasti překrytí nového bloku s částí patřící příslušnému bloku na opačné straně textury. Na obrázku 29 jsou barevně označeny oblasti, které jsou umísťovány do posledního řádku a posledního sloupce.

V případě metody Graphcuts není výběr umístění nových bloku předem daný a proto nelze postupovat stejným způsobem jako u metody Image Quilting.



Obrázek 29: Úprava Image Qulting metody pro seamless textury.

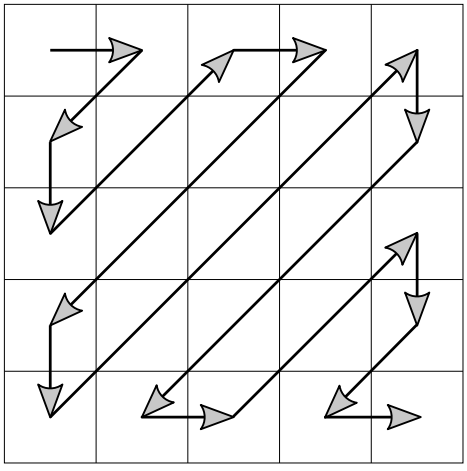
6.3 Míchání barev

U obou metod je v případě některých textur i přes výpočet minimálního řezu viditelné napojení bloků. Toto napojení je viditelné hlavně u metody Image Quilting, proto byla do implementace přidána možnost míchání barev v oblasti překrytí založená na minimálním řezu popsaná v kapitole 3.4. Ve většině případů takto použité míchání barev zlepšuje výsledky syntézy, ale v případech, kdy je zvolena příliš velká oblast překrytí bloků, dochází k přílišnému rozmazání textury. Tento problém je možné vyřešit určením, přes jak velkou část překrývající oblasti je míchání barev prováděno. Případně využitím jiného způsobu míchání barev.

Při použití metody Graphcuts nejsou ve většině případů řezy viditelné. Pro případy, kdy jsou viditelné, by bylo možné provést míchání barev podobným způsobem jako pro metodu Image Quilting.

6.4 Zrychlení

Pokud je pro syntézu použita vzorová textura o velkém rozlišení například 800x800 pixelů trvá syntéza textury o rozměrech 1024x1024 pixelů metodou Image Quilting přibližně 25 minut. V případě metody Image Quilting, tak jak je popsána v kapitole 3, není možné urychlení paralelizací výpočtů kvůli rastrovému průchodu. Při rastrovém průchodu je pro umístění nového bloku vždy potřeba čekat na umístění předešlého bloku. Tento problém lze obejít upravením pořadí, v jakém jsou bloky vkládány do syntetizované textury, na pořadí ukázané na obrázku 30. Při takto zvoleném pořadí procházení bloků je možné provádět všechny výpočty a umisťování nových bloků v diagonálním směru paralelně.



Obrázek 30: Průchod pro umisťování nových bloků v metodě Image Quilting umožňující paralelního výběru bloků.

Závěr

Vytvořená aplikace umožňuje syntézu textur pomocí tří metod. Konkrétně metodou Image Quilting, Graphcuts a upravenou metodou Image Quilting využívající detekci hran v obraze. V prvních kapitolách práce jsou popsány textury a způsob jejich mapování na povrch objektu. Následně jsou popsány základní algoritmy, které jsou používány pro syntézu textur. Poté jsou teoreticky popsány metody, které byly vybrány pro implementaci do aplikace. Po teoretické části jsou porovnány výsledky syntézy jednotlivých metod.

Při porovnávání metod bylo zjištěno, že pro strukturované (structured), regulární (regular) a buňkové (cellular) textury jsou výsledky metod Image Quilting a Graphcuts srovnatelné. Avšak v případě částečně strukturovaných/stochastických (semi-structured/stochastic) textur a textur obsahujících velké prvky (large features) jsou výsledky vygenerované metodou Graphcuts lepší. Vygenerované textury metodou Graphcuts obsahují méně artefaktů a opakování vzorové textury je méně znatelné než u textur vygenerovaných metodou Image Quilting.

Dále byla testována upravená metoda Image Quilting. Tato metoda dosahuje v kategoriích regulárních a strukturovaných textur u textur, kde je velmi malá variace v barvě jednotlivých prvků, lepších výsledků než zbývající dvě metody.

Součástí práce je také urychlení metody Graphcuts. Toto naimplementované urychlení je možné použít při inicializaci textury pouze v případě stochastických textur. Avšak pro fázi vylepšování textury je urychlení vhodné pro všechny kategorie.

Popsaná kvalita generování textur jednotlivých metod platí pro většinu textur. Vyskytují se však výjimky, kdy jedna z metod generuje lepší výsledky než zbývající dvě. Proto nelze říci, že textura spadající do dané kategorie bude vždy nejlépe vygenerována určitou metodou.

Poslední část práce je zaměřena na problémy vznikající při syntéze textur naimplementovanými metodami a jsou navrhnuta jejich možná řešení. Například urychlení metody Image Quilting nebo vylepšení generovaných výsledků přidáním možnosti vybrat oblast, která nemá být pro generování použita.

Literatura

[1] WIE, Li-Yi, LEFEBVRE, Sylvain, KWATRA, Vivek, TURK, Greg. State of the Art in Example-based Texture Synthesis. In: *Eurographics 2009 - State of the Art Reports* [online]. The Eurographics Association, 2009 [cit. 8.2.2017]. Dostupné z: `https://www.microsoft.com/en-us/research/wp-content/uploads/2009/03/texture_synthesis_eg09star.pdf`

[2] RUDOLPH, Carsten. Recent Developments in Example-based Texture Synthesis for Graphics Rendering. In: *Informatik 2016, 46. Jahrestagung der Gesellschaft Informatik, 26.-30. September 2016, Klagenfurt, GI-Edition*. Bonner Köllen Verlag, 2016, s. 2175-2185 [cit. 3.2.2017]. ISBN 978-3-88579-653-4. Dostupné z: `http://subs.emis.de/LNI/Proceedings/Proceedings259/2175.pdf`

[3] EFROS, Alexei, FREEMAN, William. Image Quilting for Texture Synthesis and Transfer. In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques, SIGGRAPH 2001* [online]. New York: ACM, 2001, s. 341-346 [cit. 1.2.2017]. ISBN 1-58113-374-X. Dostupné také z: `http://people.csail.mit.edu/billf/publications/Image_Quilting.pdf`

[4] KWATRA, Vivek, SCHÖDL, Arno, ESSA, Irfan, TURK, Greg, BOBICK, Aaron. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. In: *Proceeding ACM Transactions on Graphics, SIGGRAPH 2003* [online]. New York: ACM, 2003, s. 277-286 [cit. 8.2.2017]. ISBN 1-58113-709-5. Dostupné z: `http://www.cc.gatech.edu/cpl/projects/graphcuttextures/gc-final.pdf`

[5] SOJKA, Eduard, GAURA, Jan, KRUMNIKL, Michal. *Matematické základy digitálního zpracování obrazu* [online]. Ostrava: Vysoká škola báňská – Technická univerzita, 2011 [cit. 27.2.2017]. Dostupné z: `http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/digitalni_zpracovani_obrazu.pdf`

[6] ANGEL, Edward, SHREINER, Dave. *Interactive computer graphics: A top-down approach with Shader-Based OpenGL* [online]. 6. edice. Boston: Addison-Wesley, 2012 [cit. 14.3.2017]. ISBN 0-1325-4523-3. Dostupné také z: `http://faculty.mu.edu.sa/public/uploads/1415093060.9038[Interactive.Computer.Graphics%EF%BC%9AA.Top-Down.Approach.with.Shader-Based.OpenGL(6th.2011)].Edwar.pdf`

[7] ŽÁRA, Jiří, BENEŠ, Bedřich, SOCHOR, Jiří, FELKEL, Petr. *Moderní počítačová grafika* [online]. Brno: Computer Press, 2004 [cit. 14.3.2017]. ISBN 80-251-0454-0. Dostupné také z: `http://fei.doevil.cz/fei/2_rocnik/[ZS]%20IPOGR%20-%20Pocitacova%20grafika/Moderni-pocitacova-grafika.pdf`

[8] SOJKA, Eduard. *Počítačová grafika II:metody a nástroje pro zobrazování 3D scén* [online]. Ostrava, Vysoká škola báňská – Technická univerzita, 2003 [cit. 15.3.2017]. ISBN 80-248-0293-7. Dostupné z: http://mrl.cs.vsb.cz//people/sojka/pg/pocitacova_grafikaII.pdf

[9] BURDA, Pavel, HAVELEK, Radim, HRADECKÁ, Radoslava, KREML, Pavel. *Matematika I* [online]. Ostrava: Vysoká škola báňská – Technická univerzita, 2006 [cit. 16.3.2017]. ISBN 80- 248-1199-5. Dostupné z: https://www.vsb.cz/export/sites/vsb/714/.content/files/studijnimaterialy/m1.pdf

[10] ZHOU, Dongxiao. *Texture Analysis and Synthesis Using a Generic Markov-Gibbs Image Model* [online]. Auckland, 2006. Disertační práce. The University of Auckland. Dostupné z: https://www.cs.auckland.ac.nz/~georgy/research/texture/thesis-html/thesis.html

[11] KASPAR, Alexandre, NEUBERT, Boris, LISCHINSKI, Dani, PAULY, Mark, KOPF, Johannes. Self Tuning Texture Optimization. In: *Computer Graphics Forum, vol. 34* [online]. The Eurographics Association and John Wiley & Sons Ltd., 2015, s. 349-359 [cit. 6.2.2017]. Dostupné také z: http://www.cs.huji.ac.il/~danix/publications/SelfTuning-eg2015.pdf

[12] EFROS, Alexei, LEUNG, Thomas. Texture Synthesis by Non-parametric Sampling. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision* [online]. IEEE, 1999, s. 1033-1038 [cit. 10.3.2017]. ISBN 0-7695-0164-8. Dostupné také z: http://www.cs.ubc.ca/~little/cpsc425/efrosLeung99.pdf

[13] WEI, Li-Yi, LEVOY, Marc. Fast Texture Synthesis using Tree-structured Vector Quantization. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques, SIGGRAPH 2000* [online]. New York: ACM Press/Addison-Wesley Publishing Co., 2000, s. 479-488 [cit. 10.3.2017]. ISBN 1-58113-208-5. Dostupné z: https://graphics.stanford.edu/papers/texture-synthesis-sig00/texture.pdf

[14] LIANG, Lin, LIU, Ce, XU, Ying-Qing, GUO, Baining, SHUM, Heung-Yeung. Real-Time Texture Synthesis by Patch-Based Sampling. In: *ACM Transactions on Graphics, volume 20, issue 3, july 2001* [online]. New York: ACM, 2001, s. 127-150 [cit. 5.3.2017]. Dostupné z: https://people.csail.mit.edu/celiu/pdfs/TOG.pdf

[15] PRAUN, Emil, FINKELSTEIN, Adam, HOPPE, Hugues. Lapped Textures. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques, SIGGRAPH 2000* [online]. New York: ACM Press/Addison-Wesley Publishing Co., 2000, s. 465-470 [cit. 5.3.2017]. ISBN 1-58113-208-5. Dostupné z: http://hhoppe.com/lapped.pdf

[16] KWATRA, Vivek, ESSA, Irfan, BOBICK, Aaron, KWATRA, Nipun. Texture Optimization for Example-based Synthesis. In: *ACM Transactions on Graphics – Proceedings of ACM SIGGRAPH 2005* [online]. NEW York: ACM, 2005, s. 795-802 [cit. 8.3.2017]. Dostupné z: http://www.cc.gatech.edu/cpl/projects/textureoptimization/TO-final.pdf

[17] Weisstein, Eric, 2017. L^2-Norm. *Wolfram MathWorld* [online]. Wolfram Research, [cit. 9.2.2017]. Dostupné z: http://mathworld.wolfram.com/L2-Norm.html

[18] BRADSKI, Gary, KAEHLER, Adrian. *Learning OpenCV* [online]. Farnham: O'Reilly Media, 2008 [cit. 14.3.2017]. ISBN 978-0-596-51613-0. Dostupné také z: http://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf

[19] HUAMÁN, Ana. Sobel Derivates. In: *OpenCV Tutorials: Image Processing* [online]. OpenCV.org ©2011-2017 [cit. 17.2.2017]. OpenCV Documentation. Dostupné z: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html

[20] HUAMÁN, Ana. Canny Edge Detector. In: *OpenCV Tutorials: Image Processing* [online]. OpenCV.org ©2011-2017 [cit. 17.2.2017]. OpenCV Documentation. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

[21] HUAMÁN, Ana. Smoothing Images. In: *OpenCV Tutorials: Image Processing* [online]. OpenCV.org ©2011-2017 [cit. 18.2.2017]. OpenCV Documentation. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[22] VIOLA, Paul, JONES, Michael. Rapid Object Detection using a Boosted Cascade of Simple Features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* [online]. IEEE, 2001, s. 511-518 [cit. 21.2.2017]. ISBN 0-7695-1272-0. Dostupné také z: http://cse.msu.edu/~cse803/Readings/S4_violaJones_CVPR2001.pdf

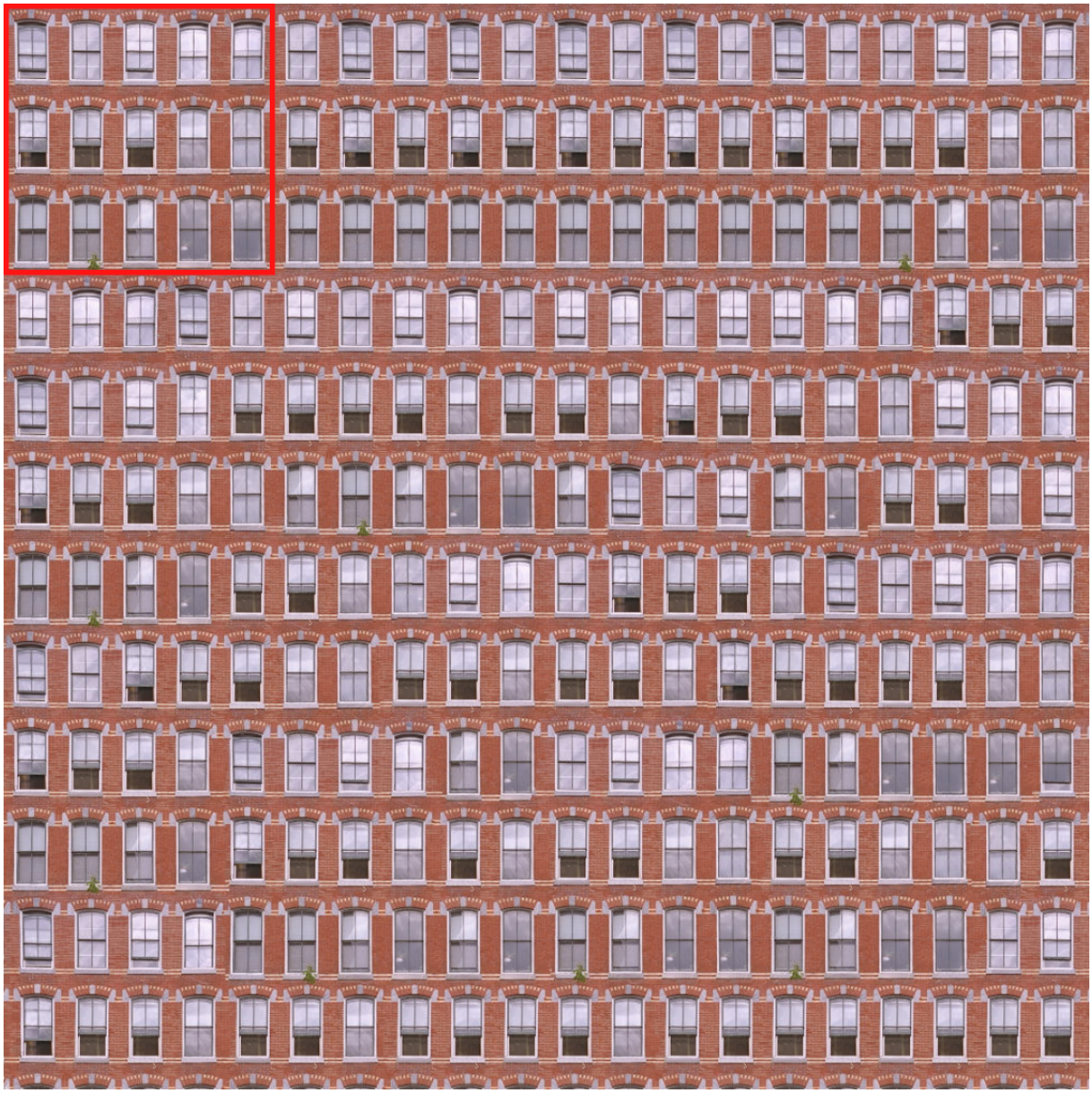
[23] BOYKOV, Yuri, VEKSLER, Olga, ZABIH, Ramin. Fast Approximate Energy Minimization via Graph Cuts. In: *Computer Vision 1999. The Proceedings of the Seventh IEEE International Conference on Computer Vision* [online]. IEEE, 1999, s. 377-384 [cit. 10.2.2017]. ISBN 0-7695-0164-8. Dostupné také z: http://www.cs.cornell.edu/rdz/papers/bvz-iccv99.pdf

[24] BOYKOV, Yuri, KOLMOGOROV, Vladimir. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004* [online]. IEEE, 2004, s. 1124-1137 [cit. 16.2.2017]. ISSN 0162-8828. Dostupné také z: `http://www.csd.uwo.ca/~yuri/Papers/pami04.pdf`

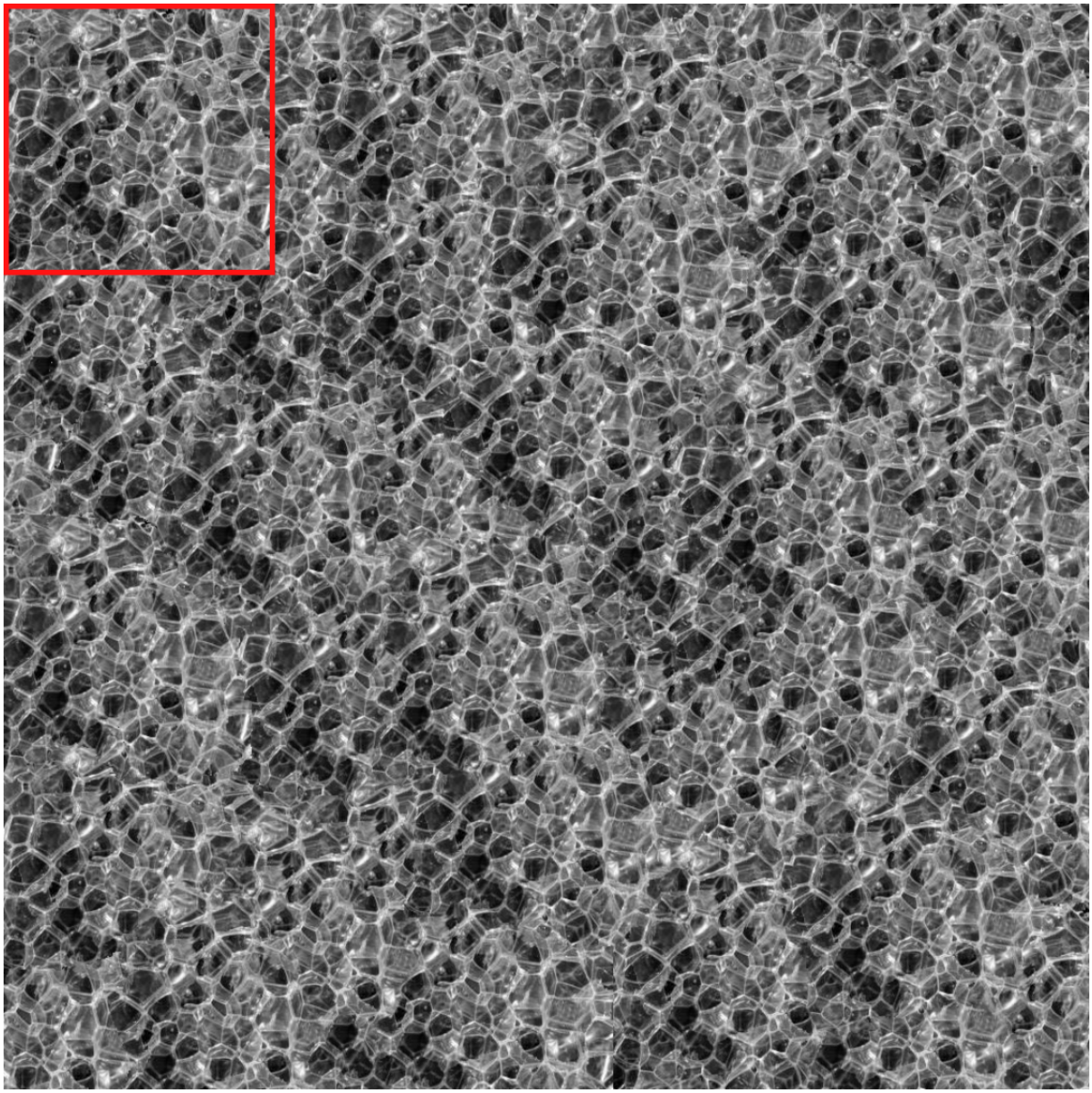
[25] KOVÁŘ, Petr. *Úvod do teorie grafů* [online]. Ostrava: Vysoká škola báňská – Technická univerzita, 2012 [cit. 10.2.2017]. Dostupné z: `http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/uvod_do_teorie_grafu.pdf`

Příloha

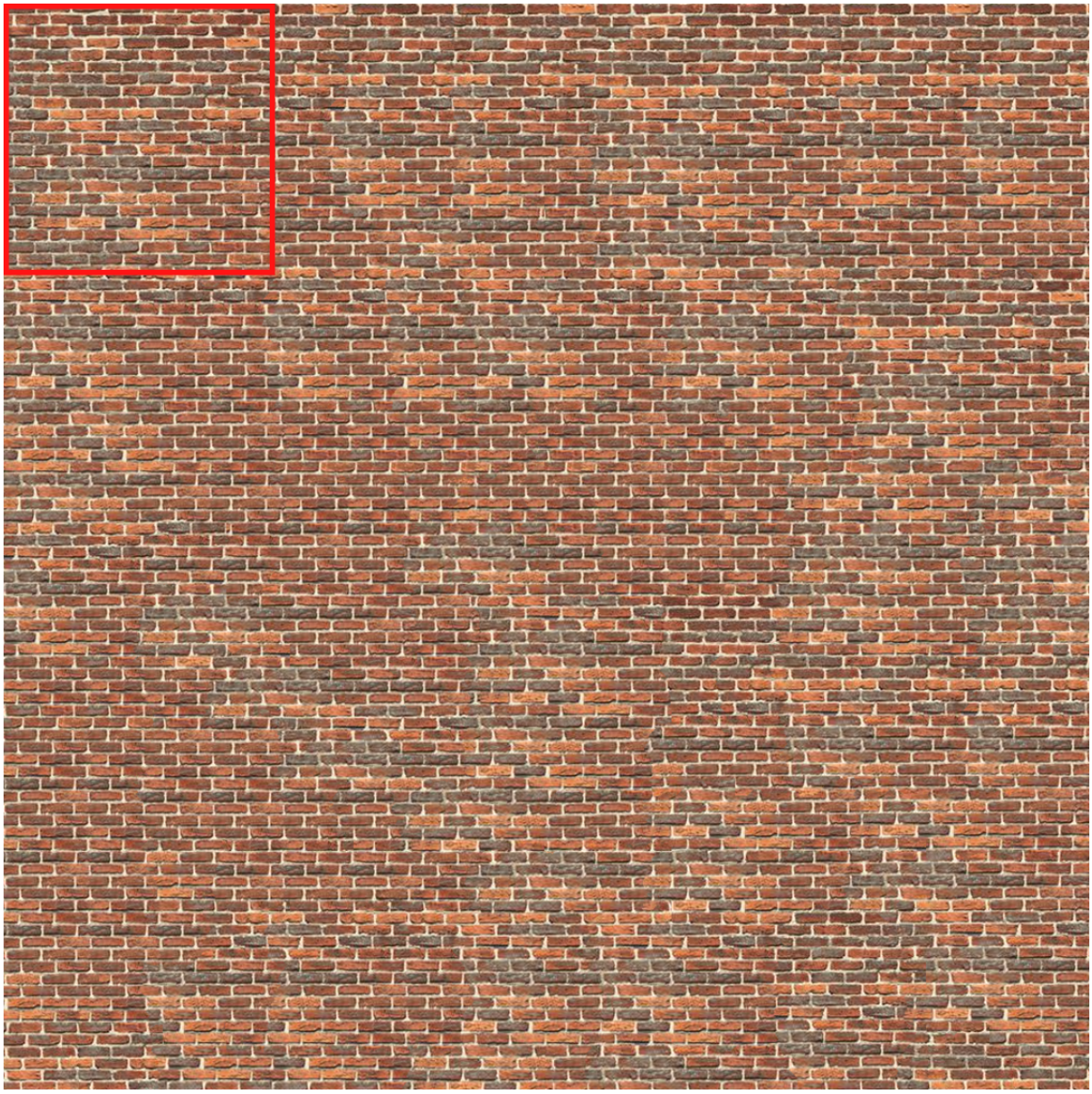
A Textura vygenerovaná metodou Graphcuts (SSD výpočet)



B Textura vygenerováná metodou Graphcuts (FFT výpočet)



C Textura vygenerovaná metodou Image Quilting



D Porovnání použití vygenerované textury a opakování vzorové textury